```
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS
SSS              YYY         YYY  SSS
SSS              YYY         YYY  SSS
SSS              YYY         YYY  SSS
SSS                 YYY   YYY     SSS
SSS                 YYY   YYY     SSS
SSS                 YYY   YYY     SSS
   SSSSSSSSS           YYY           SSSSSSSSS
   SSSSSSSSS           YYY           SSSSSSSSS
   SSSSSSSSS           YYY           SSSSSSSSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS
```

```
_$
Ps
--
YZ
Z$
Z$
Z$
Z$
Z$
Z$
Z$
Z$
Z$
Z$
Z$
Z$
```

```
**FILE**ID**SYSGETLKI                                        K  1



  SSSSSSSS  YY      YY    SSSSSSSS    GGGGGGG   EEEEEEEEEE  TTTTTTTTTT  LL          KK      KK    IIIIII
  SSSSSSSS  YY      YY    SSSSSSSS    GGGGGGG   EEEEEEEEEE  TTTTTTTTTT  LL          KK      KK    IIIIII
SS          YY      YY  SS          GG          EE              TT      LL          KK      KK      II
SS          YY      YY  SS          GG          EE              TT      LL          KK      KK      II
SS            YY  YY    SS          GG          EE              TT      LL          KK    KK        II
SS            YY  YY    SS          GG          EE              TT      LL          KK    KK        II
  SSSSSS        YY        SSSSSS    GG          EEEEEEEE        TT      LL          KKKKKK          II
  SSSSSS        YY        SSSSSS    GG          EEEEEEEE        TT      LL          KKKKKK          II
        SS      YY              SS  GG  GGGGGG  EE              TT      LL          KK    KK        II
        SS      YY              SS  GG  GGGGGG  EE              TT      LL          KK    KK        II
        SS      YY              SS  GG      GG  EE              TT      LL          KK      KK      II          ....
        SS      YY              SS  GG      GG  EE              TT      LL          KK      KK      II          ....
SSSSSSSS        YY      SSSSSSSS      GGGGGG    EEEEEEEEEE      TT      LLLLLLLLLL  KK      KK    IIIIII        ....
SSSSSSSS        YY      SSSSSSSS      GGGGGG    EEEEEEEEEE      TT      LLLLLLLLLL  KK      KK    IIIIII        ....


LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

SYS
V04

```
0000          .TITLE  SYSGETLKI - GET LOCK MANAGER INFORMATION SYSTEM SERVICE
0000          .IDENT  'V04-000'
0000    ;
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000
0000    ;++
0000    ; FACILITY: VMS Executive, System services.
0000    ;
0000    ; ABSTRACT:
0000    ;
0000    ;       Return system/cluster lock manager information.
0000    ;
0000    ; ENVIRONMENT: Kernel Mode
0000    ;
0000    ; AUTHOR: Rod N. Gamache,        CREATION DATE: 15-November-1982
0000    ;
0000    ; MODIFIED BY:
0000    ;
0000    ;       V03-014 RNG0014         Rod N. Gamache          3-Aug-1984
0000    ;               Make all Lock waiting states map to LKI$C_WAITING.
0000    ;
0000    ;       V03-013 RNG0013         Rod N. Gamache          24-Jul-1984
0000    ;               Stall access to lock database if cluster is re-configuring,
0000    ;               call lock manager routine to perform stall operation.
0000    ;
0000    ;       V03-012 RNG0012         Rod N. Gamache          01-May-1984
0000    ;               Restore the PCB address on successive loops through
0000    ;               the main proccess code, when doing a wildcard search.
0000    ;
0000    ;       V03-011 RNG0011         Rod N. Gamache          26-Mar-1984
0000    ;               Fix invalid REMLKID that is returned on Local copy LOCKS.
0000    ;
0000    ;       V03-010 RNG0010         Rod N. Gamache          21-Mar-1984
0000    ;               Return correct EPID value, return 2 more longwords in the list
0000    ;               items (REMLKID & remCSID). Set size of individual items in
0000    ;               list requests.
```

```
0000    ;                     Return SS$_IVMODE on access mode violations.
0000    ;
0000    ;       V03-009 CWH3009          CW Hobbs                  28-Feb-1984
0000    ;               Change IPL synchronization so that $GETLKI can be called
0000    ;               at IPL <= IPL$_ASTDEL.  This lets $GETDVI interrogate
0000    ;               the XQP's lock value block so that $GETDVI can return
0000    ;               the correct value for DVI$_FREEBLOCKS.
0000    ;
0000    ;       V03-008 RNG0008          Rod N. Gamache            05-Dec-1983
0000    ;               Change references to LOCK STRUCTURES to reflect changes made
0000    ;               in the Lock Manager.
0000    ;
0000    ;       V03-007 RNG0007          Rod N. Gamache            07-Oct-1983
0000    ;               Fix synchronization problem caused by exec routine that
0000    ;               lowers IPL; wrote inline code to replace exec routine.
0000    ;
0000    ;       V03-006 CWH3006          CW Hobbs                  23-Sep-1983
0000    ;               Fix broken branch
0000    ;
0000    ;       V03-005 RNG0005          Rod N. Gamache            31-Aug-1983
0000    ;               Deliver AST's only on success.
0000    ;               Allow EXEC mode and KERNEL mode users access to system locks.
0000    ;               Return zero REMLKID if CSID is zero.
0000    ;
0000    ;       V03-004 RNG0004          Rod N. Gamache            05-Aug-1983
0000    ;               Add REMLKID item code.
0000    ;               Return SS$_NOMORELOCK error instead of SS$_NOMOREPROC.
0000    ;               Add support for distributed list items (LOCKS, BLOCKEDBY
0000    ;               and BLOCKING).
0000    ;               Make sure user has sufficient BYCNT quota for list operations.
0000    ;               Return proper CSID in the event the CSID of the RSB is zero.
0000    ;
0000    ;       V03-003 RNG0003          Rod N. Gamache            05-May-1983
0000    ;               Return "external" PID wherever necessary. Return
0000    ;               SS$_NOWORLD error instead of SS$_NOPRIV.
0000    ;               Add support for distributed GETLKI.
0000    ;
0000    ;       V03-002 SRB0073          Steve Beckhardt           30-Mar-1983
0000    ;               Fix broken ASSUME statement.
0000    ;
0000    ;       V03-001 RNG0001          Rod N. Gamache            14-Mar-1983
0000    ;               Remove SYSNAM bit from RMOD field. Change RMOD to be a
0000    ;               full byte. Use RMOD in RSB rather than LKB.
0000    ;--
```

```
                 0000
                 0000   ;
                 0000   ; EQUATED SYMBOLS:
                 0000   ;
                 0000
        00000002 0000           MAXSTRUC = 2                            ; Maximum number of structures
                 0000
        00000004 0000           EFN = 4                                 ; event flag number argument
        00000008 0000           LKID = 8                                ; address of the lock ID
        0000000C 0000           ITMLST = 12                             ; address of item identifiers
        00000010 0000           IOSB = 16                               ; I/O status block address
        00000014 0000           ASTADR = 20                             ; ast routine address
        00000018 0000           ASTPRM = 24                             ; ast parameter
        0000001C 0000           RESERV = 28                             ; RESERVED
                 0000
                 0000   ;
                 0000   ; One quadword local is left on stack for routines which may
                 0000   ; manipulate values before returning them.
                 0000   ;
                 0000
        FFFFFFF8 0000           LOCAL_SPACE = -8
        FFFFFFFC 0000           SAVED_IPL   = -4                        ; We will reference stored IPL off the frame
                 0000
        00000005 0000           MAX_LKB_ITEM = <LKI$_LASTLKB&^XFF>-1 ; maximum LKBTBL item number
        00000008 0000           MAX_RSB_ITEM = <LKI$_LASTRSB&^XFF>-1 ; maximum RSBTBL item number
                 0000
                 0000   ;
                 0000   ; Data type codes (all numeric types have same code)
                 0000   ;
                 0000
        00000000 0000           VALUE = 0                               ; numeric value
        00000001 0000           BSTRING = 1                             ; blank filled string
        00000002 0000           CSTRING = 2                             ; counted ascii string
                 0000   ;
                 0000   ; AST control block extensions
                 0000   ;
                 0000           $DEFINI ACB
                 0000
        0000001C 0000           .=ACB$L_KAST+4                                  ;
                 001C
                 001C   $DEF    ACB_L_DADDR     .BLKL   1                       ; data buffer address
                 0020   $DEF    ACB_L_EFN       .BLKL   1                       ; event flag number
                 0024   $DEF    ACB_L_IOSB      .BLKL   1                       ; completion AST routine addr
                 0028   $DEF    ACB_L_OPID      .BLKL   1                       ; original requester's PID
                 002C   $DEF    ACB_L_COUNT     .BLKL   1                       ; item descriptor count
                 0030   $DEF    ACB_L_ILIST                                     ; item descriptor list
                 0030
        0000000C 0030           ACB_C_IDESC = 12                                ; item descriptor size
                 0030
                 0030           $DEFEND ACB
                 0000
                 0000
                 0000   ;
                 0000   ; OWN STORAGE:
                 0000   ;
                 0000
             00000000           .PSECT  WSYSGETLKI                              ; Non-paged PSECT
```

D 2

```
                 0000
                 0000
                 0000   ;
                 0000   ; This array contains the maximum item number for both of the item
                 0000   ; data structures, indexed by structure number.
                 0000   ;
                 0000
                 0000   MAXCOUNT:
             05  0000           .BYTE   MAX_LKB_ITEM
             08  0001           .BYTE   MAX_RSB_ITEM
                 0002
                 0002   ;
                 0002   ; The tables contain a word offset followed by a byte code for each item
                 0002   ; followed by a byte of structure type.  The code contains the length of
                 0002   ; the item in the low five bits, and the item type in the high three bits.
                 0002   ; The types are value, counted string, and blank filled string.
                 0002   ;
                 0002
                 0002   LKBTBL:
       00000026  0002           .BLKB   6*<MAX_LKB_ITEM+1>                      ; define LKB table
                 0026   RSBTBL:
       0000005C  0026           .BLKB   6*<MAX_RSB_ITEM+1>                      ; define RSB table
                 005C
                 005C           .SAVE                                           ; save current location counter
                 005C
                 005C   ;
                 005C   ; Define entries to LKBTBL
                 005C   ;
       00000000  005C           LIMSGSK_ZERO = 0                                ; Define empty holder
                 005C
                 005C           LKBITM  PID,L_EPID,VALUE,4                      ; EPID of owner process
                 0008           LKBITM  LCKREFCNT,W_REFCNT,VALUE,2              ; sub-lock reference count
                 001A           LKBITM  STATE,B_RQMODE,VALUE,3,L_STATE          ; current state of lock
                 000E           LKBITM  PARENT,L_PARENT,VALUE,4                 ; LOCK ID of parent lock
                 0014           LKBITM  LOCKID,L_LKID,VALUE,4                   ; LOCK ID of lock
                 0020           LKBITM  REMLKID,L_REMLKID,VALUE,4               ; Remote LOCK ID of lock
                 0026
                 0026   ;
                 0026   ; Define entries to RSBTBL
                 0026   ;
                 0026
                 0026           RSBITM  RESNAM,B_RSNLEN,CSTRING,31              ; resource name
                 0032           RSBITM  RSBREFCNT,W_REFCNT,VALUE,2,L_RSBREFCNT ; sub-resource reference coun
                 0038           RSBITM  VALBLK,Q_VALBLK,BSTRING,16,Q_VALBLK ; value block
                 003E           RSBITM  SYSTEM,L_CSID,VALUE,4                   ; system id of system which has
                 0044                                                           ;  the master copy of resource
                 0044           RSBITM  NAMSPACE,W_GROUP,VALUE,4                ; resource name space
                 002C           RSBITM  LCKCOUNT,L_GRQFL,VALUE,4,L_LCKCOUNT ; count of locks granted on reso
                 004A           RSBITM  BLOCKEDBY,L_GRQFL,VALUE,-               ; list of locks blocked by LKID
                 004A                   LKI$C_LENGTH
                 0050           RSBITM  BLOCKING,L_WTQFL,VALUE,-                ; list of locks blocking LKID
                 0050                   LKI$C_LENGTH
                 0056           RSBITM  LOCKS,L_GRQFL,VALUE,LKI$C_LENGTH ; list of associated locks
                 005C
                 005C
       0000005C  005C           .RESTORE                                        ; restore location counter
                 005C
             005C  ;
             005C  ; Table to define items which must be handled
             005C  ; by action routines.
             005C  ;
             005C
             005C  SPECIAL:
             005C          SPECIAL_ITEM    PID,SPC_PID               ; PID of owner process
             0062          SPECIAL_ITEM    STATE,SPC_STATE           ; current state of lock
             0068          SPECIAL_ITEM    PARENT,SPC_PARENT         ; LOCK ID of parent lock
             006E          SPECIAL_ITEM    SYSTEM,SPC_SYSTEM         ; CSID of master
             0074          SPECIAL_ITEM    NAMSPACE,SPC_NAMSPACE     ; resource name space
             007A          SPECIAL_ITEM    LCKCOUNT,SPC_LCKCOUNT     ; count of locks granted on resource
             0080          SPECIAL_ITEM    BLOCKEDBY,SPC_BLOCKEDBY   ; list of locks blocked by LKID
             0086          SPECIAL_ITEM    BLOCKING,SPC_BLOCKING     ; list of locks blocking LKID
             008C          SPECIAL_ITEM    LOCKS,SPC_LOCKS           ; list of associated locks
             0092          SPECIAL_ITEM    REMLKID,SPC_REMLKID       ; Remote lock id
             0098
    0000000A 0098  SPECIAL_LEN = <.-SPECIAL>/6                       ; compute number of entries
```

```
0098          .SBTTL  SYSGETLKI - GETLKI get lock manager information system service
0098
0098   ;++
0098   ;
0098   ; FUNCTIONAL DESCRIPTION:
0098   ;
0098   ;       This service allows a process to receive information about the
0098   ;       locks, or any process locks which it has the privilege to examine.
0098   ;
0098   ; CALLING SEQUENCE:
0098   ;
0098   ;       CALLS/CALLG
0098   ;
0098   ;               Actually, this routine MUST be called through the system
0098   ;               service dispatcher.
0098   ;
0098   ; INPUTS:
0098   ;
0098   ;       R4                      PCB address of requesting process
0098   ;
0098   ;       EFN(AP)                 number of the event flag to set when all of the
0098   ;                               requested data is valid.
0098   ;       LKID(AP)                address of a longword containing the process ID of the
0098   ;                               process for which the information is being requested
0098   ;       ITMLST(AP)              address of a list of item descriptors of the form:
0098   ;
0098   ;               +--------------------------------+
0098   ;               !  ITEM CODE     !  BUF. LENGTH  !
0098   ;               +--------------------------------+
0098   ;               !         BUFFER ADDRESS         !
0098   ;               +--------------------------------+
0098   ;               !   ADDRESS TO RETURN LENGTH     !
0098   ;               +--------------------------------+
0098   ;
0098   ;       IOSB(AP)                address of a quadword I/O status block to receive final
0098   ;                               status
0098   ;       ASTADR(AP)              address of an AST routine to be called when all of the
0098   ;                               requested data has been supplied.
0098   ;       ASTPRM(AP)              32 bit ast parameter
0098   ;
0098   ; IMPLICIT INPUTS:
0098   ;
0098   ;       IPL <= IPL$_ASTDEL              This allows other system services which are
0098   ;                                       holding mutexes to call $GETLKI.
0098   ;
0098   ; OUTPUTS:
0098   ;
0098   ;       none
0098   ;
0098   ; IMPLICIT OUTPUTS:
0098   ;
0098   ;       none
0098   ;
0098   ; ROUTINE VALUE:
0098   ;
0098   ;       SS$_NORMAL              normal completion.
0098   ;       SS$_ACCVIO              ITMLST can not be read by the calling access mode.
```

```
                            0098  ;                           or the return buffer or return length word can not
                            0098  ;                           be written by the calling access mode.
                            0098  ;           SS$_BADPARAM    an invalid item identifier was supplied.
                            0098  ;           SS$_IVLOCKID    lock id specified is not valid.
                            0098  ;           SS$_BUFFEROVF   data has overflowed the user buffer.
                            0098  ;           SS$_NOSYSLCK    SYSLCK privilege is needed to access this information.
                            0098  ;           SS$_NOWORLD     WORLD privilege is needed to access this information.
                            0098  ;           SS$_EXQUOTA     User's AST or BYTLM quota has been exceeded.
                            0098  ;           SS$_INSFMEM     Insufficient non-paged dynamic memory.
                            0098  ;           SS$_NOMORELOCK  No more locks on "wildcard" search operation.
                            0098  ;
                            0098  ; SIDE EFFECTS:
                            0098  ;
                            0098  ;           none
                            0098  ;--
                            0098
                        00000000          .PSECT  YEXEPAGED               ; Only entry mask in this program section
                            0000
                      0FFC  0000          .ENTRY  EXE$GETLKI,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
          00000098'EF   17  0002          JMP     EXE_GETLKI              ; Transfer to real procedure
                            0008
                        00000098          .PSECT  WSYSGETLKI
                            0098
                            0098  EXE_GETLKI:
                            0098          DSBINT  #IPL$_SYNCH             ; Raise IPL to check lock mgr database
          00000000'GF   16  009E          JSB     G^LCK$CHECK_STALL       ; ... stall if needed (in CALLER's mode)
                            00A4          SETIPL  #IPL$_ASTDEL            ; Set IPL to ASTDEL
       5E    F8 AE      DE  00A7          MOVAL   LOCAL_SPACE(SP),SP      ; Allocate local space on stack
                5B      D4  00AB          CLRL    R11                     ; Assume no remote LOCK information
 54       00000000'EF   D0  00AD  2$:     MOVL    SCH$GL_CURPCB,R4        ; Reset PCB address
          50    5B      D0  00B4          MOVL    R11,R0                  ; Any remote LOCK BLOCK?
                08      13  00B7          BEQL    3$                      ; Br if not, okay
                5B      D4  00B9          CLRL    R11                     ; No more remote lock block
          00000000'EF   16  00BB          JSB     EXE$DEANONPAGED         ; Else, deallocate the remote lock block
              03CF      30  00C1  3$:     BSBW    GETLKB                  ; Get LKB address of desired lock
             5C 50      E9  00C4          BLBC    R0,17$                  ; Exit if invalid LKID specified
                            00C7  ;
                            00C7  ; Check for, and clear event flag
                            00C7  ;
       53    04 AC      9A  00C7          MOVZBL  EFN(AP),R3              ; Get event flag number
          00000000'EF   16  00CB          JSB     SCH$CLREF               ; Clear this event flag
             4F 50      E9  00D1          BLBC    R0,17$                  ; And return on errors.
                            00D4  ;
                            00D4  ; Check for, and clear possible IOSB
                            00D4  ;
       51    10 AC      D0  00D4          MOVL    IOSB(AP),R1             ; Get IOSB address
                0B      13  00D8          BEQL    5$                      ; Branch if none
                            00DA          IFWRT   #8,(R1),4$              ; Check access to it
              0082      31  00E0          BRW     30$                     ; Else, return error
                61      7C  00E3  4$:     CLRQ    (R1)                    ; Clear IOSB
                            00E5  ;
                            00E5  ; Validate AST, if present. Note R4 still has our PCB address, and R9
                            00E5  ; has the LKB address of the lock we want information from.
                            00E5  ;
             14 AC      D5  00E5  5$:     TSTL    ASTADR(AP)
                05      13  00E8          BEQL    7$                      ; No AST to check.
             38 A4      B5  00EA          TSTW    PCB$W_ASTCNT(R4)        ; Is quota exceeded?
```

```
           7B   15  00ED              BLEQ    35$                     ; Branch if so and return error
                    00EF      ;
                    00EF      ; Check if information is contained on another system in the cluster
                    00EF      ;
         00DE   30  00EF      7$:     BSBW    GET_REMLKI              ; Get remote LKI block if needed
        2E 50   E9  00F2              BLBC    R0,T7$                  ; Exit on error
                    00F5      ;
                    00F5      ; Loop through the item descriptor blocks, validating the requested item
                    00F5      ; identifiers and moving accessible items.  A zero item identifier terminates
                    00F5      ; the list.
                    00F5      ;
                    00F5      ; At this point:
                    00F5      ;
                    00F5      ;       R4 = PCB address
                    00F5      ;       R9 = LKB address
                    00F5      ;       R11 = Remote lock block information or zero
                    00F5      ;       AP = Pointer to argument list
                    00F5
   55    0C AC  D0  00F5      10$:    MOVL    ITMLST(AP),R5           ; Get item descriptor list address
                    00F9              IFNORD  #4,(R5),30$             ; Check first longword readable
      56    85  3C  00FF      15$:    MOVZWL  (R5)+,R6                ; Get buffer size
      51    85  3C  0102              MOVZWL  (R5)+,R1                ; Get item identifier
            6D  13  0105              BEQL    50$                     ; Done if zero, take normal exit
                    0107              IFNORD  #12,(R5),30$            ; Check rest of this descriptor ...
                    010D                                              ; ... plus first longword of next one
      57    85  7D  010D              MOVQ    (R5)+,R7                ; Get buffer address and return address
            51  DD  0110              PUSHL   R1                      ; Save R1 across accessibility check
      50    57  D0  0112              MOVL    R7,R0                   ; Buffer address to R0
      51    56  D0  0115              MOVL    R6,R1                   ; And size to R1
            53  D4  0118              CLRL    R3                      ; PROBE will use PSL<PRVMOD>
   00000000'EF  16  011A              JSB     EXE$PROBEW              ; Check write accessibility of buffer
         51   8ED0  0120              POPL    R1                      ; Restore R1 for use by CHECKITEM
      51    50  E9  0123      17$:    BLBC    R0,GRET                 ; Return error if inaccessible
                    0126              ;
                    0126              ; We will raise IPL to IPL$_SYNCH to lock down the LKB. We will
                    0126              ; have to verify that the LKB is still valid, before proceeding.
                    0126              ;
                    0126              ; The IPL will be restored by the MOVEIT routine just before copying
                    0126              ; the data to the users's buffer. This is done to allow the SPC_xxx
                    0126              ; routines to gather up any additional information that needs to be
                    0126              ; returned to the user, without verifying that the LKB address is
                    0126              ; still valid.
                    0126              ;
                    0126              SETIPL  #IPL$_SYNCH             ; Raise IPL to sync access to structures
                    0129                                              ;  can't reference user's process space
   54    30 A9  3C  0129              MOVZWL  LKB$L_LKID(R9),R4       ; Get lock index
   00000000'EF  54  D1  012D          CMPL    R4,LCK$GL_MAXID         ; Is the lock index still ok?
            0A  1A  0134              BGTRU   20$                     ; Br if no, check for error condition
 00000000'FF44  59  D1  0136          CMPL    R9,@LCK$GL_IDTBL[R4]    ; Is the lock address still the same?
            12  13  013E              BEQL    25$                     ; Br if yes, okay to proceed
                    0140      20$:    SETIPL  #IPL$_ASTDEL            ; Restore the IPL on error condition
         08 BC  D5  0143              TSTL    @LKID(AP)               ; Is this a "wildcard" search?
            03  14  0146              BGTR    23$                     ; Br if no, continue
          FF62  31  0148              BRW     2$                      ; Else, try for next lock
  50   2124 8F  3C  014B      23$:    MOVZWL  #SS$_IVLOCKID,R0        ; Invalid lock id
            25  11  0150              BRB     GRET                    ; Return to user
                    0152              ;
```

```
                      0152          ; Check item code and return the info to user.
                      0152          ;
              55  DD  0152    25$:  PUSHL   R5                      ; Save R5 from action routines
            009C  30  0154          BSBW    CHECKITEM               ; Validate identifier and get item info.
           15 50  E9  0157          BLBC    R0,40$                  ; Invalid item if error
            00FD  30  015A          BSBW    MOVEIT                  ; Move item to user
                      015D                                          ;  NOTE: IPL is restored to IPL$_ASTDEL
            55 8ED0   015D          POPL    R5                      ; Restore R5
           9C 50  E8  0160          BLBS    R0,15$                  ; Back for next descriptor if ok
              12  11  0163          BRB     GRET                    ; Else, return error
                      0165
       50     0C  3C  0165    30$:  MOVZWL  #SS$_ACCVIO,R0          ; Access violation
              0D  11  0168          BRB     GRET
                      016A
       50     1C  3C  016A    35$:  MOVZWL  #SS$_EXQUOTA,R0         ; AST quota exceeded
              08  11  016D          BRB     GRET
                      016F
       50     14  3C  016F    40$:  MOVZWL  #SS$_BADPARAM,R0        ; Illegal item or request
              03  11  0172          BRB     GRET
                      0174
       50     01  3C  0174    50$:  MOVZWL  #SS$_NORMAL,R0          ; Normal return
                      0177
                      0177    ;
                      0177    ; Set the event flag, post the completion status, and declare a completion AST
                      0177    ;
              50  DD  0177  GRET:   PUSHL   R0                      ; Save completion status
       50     5B  D0  0179          MOVL    R11,R0                  ; Any remote lock block?
              06  13  017C          BEQL    5$                      ; Br if not, okay
       00000000'EF 16 017E          JSB     EXE$DEANONPAGED         ; Else, deallocate the remote lock block
                      0184    5$:   SETIPL  SAVED_IPL(FP)           ; Restore IPL to that on entry to service
   54  00000000'EF D0 0188          MOVL    SCH$GL_CURPCB,R4        ; Get PCB address
        51    60 A4 D0 018F         MOVL    PCB$L_PID(R4),R1        ; Get process's PID
              52  D4  0193          CLRL    R2                      ; Set null priority increment
        53    04 AC D0 0195         MOVL    EFN(AP),R3              ; Get event flag number to set
       00000000'EF 16 0199          JSB     SCH$POSTEF              ; Set the event flag
        51    10 AC D0 019F   10$:  MOVL    IOSB(AP),R1             ; Get address of IOSB
              09  13  01A3          BEQL    20$                     ; Branch if none
                      01A5          IFNOWRT #8,(R1),20$             ; Check if writable
           61 6E  D0  01AB          MOVL    (SP),(R1)               ; Store completion status
        55    14 AC D0 01AE   20$:  MOVL    ASTADR(AP),R5           ; Get address of AST routine
              18  13  01B2          BEQL    30$                     ; Branch if none specified
           15 6E  E9  01B4          BLBC    (SP),30$                ; No completion AST on error!
              54  DC  01B7          MOVPSL  R4                      ; Get PSL
   54   54    02  16  EF  01B9      EXTZV   #PSL$V_PRVMOD,#PSL$S_PRVMOD,R4,R4 ; Extract previous mode
                      01BE          $DCLAST_S (R5),ASTPRM(AP),R4    ; Queue the completion AST
            50 8ED0   01CC    30$:  POPL    R0                      ; Restore completion status
              04      01CF          RET                             ; And return.
                      01D0
```

```
                        01D0            .SBTTL  GET_REMLKI- Get remote LKI block
                        01D0
                        01D0    ;++
                        01D0    ;
                        01D0    ; FUNCTIONAL DESCRIPTION:
                        01D0    ;
                        01D0    ;       Routine to get the remote LKI block if necessary.
                        01D0    ;
                        01D0    ; CALLING SEQUENCE:
                        01D0    ;
                        01D0    ;       JSB/BSB
                        01D0    ;
                        01D0    ; INPUTS:
                        01D0    ;
                        01D0    ;       R4      PCB address
                        01D0    ;       R9      LKB address
                        01D0    ;       R11     ZERO
                        01D0    ;
                        01D0    ; IMPLICIT INPUTS:
                        01D0    ;
                        01D0    ;       IPL = IPL$_ASTDEL
                        01D0    ;
                        01D0    ; OUTPUTS:
                        01D0    ;
                        01D0    ;       R0      success/failure of operation + special flags
                        01D0    ;       R4      PCB address
                        01D0    ;       R9      LKB address
                        01D0    ;       R11     Address of remote LKI block or zero
                        01D0    ;
                        01D0    ; IMPLICIT OUTPUTS:
                        01D0    ;
                        01D0    ;       none
                        01D0    ;
                        01D0    ; SIDE EFFECTS:
                        01D0    ;
                        01D0    ;       R0-R3,R8 destroyed.
                        01D0    ;
                        01D0    ;--
                        01D0
                        01D0            .ENABL  LSB
                        01D0    GET_REMLKI:                                     ; Get remote LKI block
         50    01   90  01D0            MOVB    #1,R0                           ; Assume success
               04   E0  01D3            BBS     #LKB$V_MSTCPY,-                 ; Br if this is the master copy,
      1A  2A   A9       01D5                    LKB$W_STATUS(R9),10$            ;  information is local to this system
      58  50   A9   D0  01D8            MOVL    LKB$L_RSB(R9),R8                ; Get RSB address
      53  38   A8   D0  01DC            MOVL    RSB$L_CSID(R8),R3               ; Is this a process copy?
               10   13  01E0            BEQL    10$                             ; Br if not, information is still local
                        01E2            SETIPL  #IPL$_SYNCH                     ; Raise IPL to SYNCH
      00000000'GF   16  01E5            JSB     G^LKI$SND_STDREQ                ; And send request for information
                        01EB                                                    ;  to remote system
54    00000000'EF   D0  01EB            MOVL    SCH$GL_CURPCB,R4                ; Get our PCB address
                    05  01F2    10$:    RSB                                     ; Return to caller
                        01F3
                        01F3            .DSABL  LSB
                        01F3
```

```
                             01F3          .SBTTL  CHECKITEM - Validate item identifier
                             01F3
                             01F3   ;++
                             01F3   ;
                             01F3   ; FUNCTIONAL DESCRIPTION:
                             01F3   ;
                             01F3   ;       Routine to validate item identifier and return information
                             01F3   ;       about the item.
                             01F3   ;
                             01F3   ; CALLING SEQUENCE:
                             01F3   ;
                             01F3   ;       JSB/BSB
                             01F3   ;
                             01F3   ; INPUTS:
                             01F3   ;
                             01F3   ;       R1      item identifier
                             01F3   ;       R9      LKB address
                             01F3   ;       R11     REMOTE LKI BLOCK or zero
                             01F3   ;
                             01F3   ; IMPLICIT INPUTS:
                             01F3   ;
                             01F3   ;       IPL = IPL$_SYNCH
                             01F3   ;
                             01F3   ; OUTPUTS:
                             01F3   ;
                             01F3   ;       R0      success/failure of operation + special flags
                             01F3   ;       R1      item identifier
                             01F3   ;       R2      structure number
                             01F3   ;       R3      item length
                             01F3   ;       R4      item address
                             01F3   ;       R5      item type code
                             01F3   ;
                             01F3   ; IMPLICIT OUTPUTS:
                             01F3   ;
                             01F3   ;       none
                             01F3   ;
                             01F3   ; SIDE EFFECTS:
                             01F3   ;
                             01F3   ;       none
                             01F3   ;
                             01F3   ;--
                             01F3
                             01F3   CHECKITEM:
                  50   D4    01F3          CLRL    R0                       ; Assume bad item code
             53   51   9A    01F5          MOVZBL  R1,R3                    ; Get item number
   52   51   08   08   EF    01F8          EXTZV   #8,#8,R1,R2              ; Get structure number
                  5A   13    01FD          BEQL    80$                      ; Error if structure number zero
             02   52   91    01FF          CMPB    R2,#MAXSTRUC             ; Structure number valid?
                  55   1A    0202          BGTRU   80$                      ; Error if not
      FDF5 CF42   53   91    0204          CMPB    R3,MAXCOUNT-1[R2]        ; Check max item values (1 origin)
                  4D   1A    020A          BGTRU   80$                      ; Error if illegal item number
                             020C          CASE    R2,<10$,30$>B,#1         ; Case on structure base
                             0214   ;
                             0214   ; LKB return item
                             0214   ;
             54   59   D0    0214   10$:   MOVL    R9,R4                    ; Get back LKB address
        55   FDE7 CF   DE    0217          MOVAL   LKBTBL,R5                ; Get address of LKB item table
                  09   11    021C          BRB     40$                      ; Continue
```

```
                                021E          ;
                                021E          ; RSB return item
                                021E          ;
        54      50 A9    D0     021E     30$: MOVL     LKB$L_RSB(R9),R4           ; Get resource block address
     55      FE00 CF     DE     0222          MOVAL    RSBTBL,R5                  ; Get address of PHD item table
                                0227
     50      53   01     78     0227     40$: ASHL     #1,R3,R0                   ; Double item number
          53    6543     DE     022B          MOVAL    (R5)[R3],R3                ; Compute address in item table
              53  50     C0     022F          ADDL     R0,R3                      ;        ...
              55  83     3C     0232          MOVZWL   (R3)+,R5                   ; Get offset into data structure
              54  55     C0     0235          ADDL     R5,R4                      ; Form complete address
              50  01     D0     0238          MOVL     #1,R0                      ; Set successful return
                  5B     D5     023B          TSTL     R11                        ; Is there a remote LKI block?
                  10     13     023D          BEQL     50$                        ; Br if not, continue
              02  A3     B5     023F          TSTW     2(R3)                      ; Is this item in remote LKI block?
                  0B     13     0242          BEQL     50$                        ; Br if not
        54    02  A3     3C     0244          MOVZWL   2(R3),R4                   ; Else, get offset in remote LKI block
              54  5B     C0     0248          ADDL     R11,R4                     ; Form complete address
          00  50  01     E2     024B          BBSS     #1,R0,50$                  ; Indicate that no special lookup needed
  55    63   03   05     EF     024F     50$: EXTZV    #5,#3,(R3),R5              ; Get item type code
  53    63   05   00     EF     0254          EXTZV    #0,#5,(R3),R3              ; Get item length
                         05     0259     80$: RSB                                 ; Return to caller
                                025A
```

```
                        025A          .SBTTL  MOVEIT - Move data to user's buffer
                        025A
                        025A    ;++
                        025A    ;
                        025A    ; FUNCTIONAL DESCRIPTION:
                        025A    ;
                        025A    ;       Move the requested data to user buffer. Zero fill to end of buffer.
                        025A    ;       Return actual data length to user.  Assumes user's buffer has
                        025A    ;       been probed.
                        025A    ;
                        025A    ; CALLING SEQUENCE:
                        025A    ;
                        025A    ;       JSB/BSB
                        025A    ;
                        025A    ; INPUTS:
                        025A    ;
                        025A    ;       R0      special lookup flag
                        025A    ;       R1      item identifier
                        025A    ;       R2      data structure number
                        025A    ;       R3      item length
                        025A    ;       R4      item address
                        025A    ;       R5      item type code
                        025A    ;       R6      user buffer length
                        025A    ;       R7      user buffer address
                        025A    ;       R8      address to return length
                        025A    ;       R9      LKB address
                        025A    ;
                        025A    ; IMPLICIT INPUTS:
                        025A    ;
                        025A    ;       IPL = IPL$_SYNCH
                        025A    ;
                        025A    ; OUTPUTS:
                        025A    ;
                        025A    ;       none
                        025A    ;
                        025A    ; IMPLICIT OUTPUTS:
                        025A    ;
                        025A    ;       IPL = IPL$_ASTDEL
                        025A    ;
                        025A    ; ROUTINE VALUE:
                        025A    ;
                        025A    ;       SS$_NORMAL      Normal successful completion
                        025A    ;       SS$_ACCVIO      Access violation on attempt to access return size
                        025A    ;
                        025A    ; SIDE EFFECTS:
                        025A    ;
                        025A    ;       Registers R1-R4 destroyed
                        025A    ;--
                        025A
                        025A    MOVEIT:
                        025A            ;
                        025A            ; Call routine to check for special conditions
                        025A            ;
           5A    D4     025A            CLRL    R10                     ; No buffer to deallocate - yet!
   02 50   01    E0     025C            BBS     #1,R0,5$                ; Br if no special lookup needed
           4B    10     0260            BSBB    CHECK_SPC               ; Check for special actions
                        0262    5$:     SETIPL  #IPL$_ASTDEL            ; Restore IPL to ASTDEL
```

```
             2E 50    E9 0265           BLBC    R0,40$                  ; Br if error
                         0268           ;
                         0268           ; Check for counted string, and find actual length if so.
                         0268           ;
             55    02 D1 0268           CMPL    #CSTRING,R5             ; Is this special string?
                   03 12 026B           BNEQ    10$                     ; Br if not
             53    84 9A 026D           MOVZBL  (R4)+,R3                ; Get length and skip length byte
                         0270           ;
                         0270           ; Move the data
                         0270           ;
                   28 BB 0270      10$: PUSHR   #^M<R3,R5>              ; Save registers
 67    56    00    64 53 2C 0272        MOVC5   R3,(R4),#0,R6,(R7)      ; Move data to user's buffer, zero fill
                   28 BA 0278           POPR    #^M<R3,R5>              ; Restore registers
                   58 D5 027A           TSTL    R8                      ; Did caller want return length?
                   15 13 027C           BEQL    30$                     ; Br if not
                         027E           IFNOWRT #4,(R8),70$             ; Br if longword not writeable
             56    53 B1 0284           CMPW    R3,R6                   ; See how much was moved
                   07 15 0287           BLEQ    20$                     ; Use valid data length if it fits
             53    56 B0 0289           MOVW    R6,R3                   ; Else give him "too short" buffer size
          00 53    1F E2 028C           BBSS    #31,R3,20$              ; And return buffer overflow indicator
             68    53 D0 0290      20$: MOVL    R3,(R8)                 ; Return length to user
             50    01 9A 0293      30$: MOVZBL  S^#SS$_NORMAL,R0        ; Set success code
                   5A D5 0296      40$: TSTL    R10                     ; Any pool deallocation needed?
                   0D 13 0298           BEQL    50$                     ; Br if no
                   0F BB 029A           PUSHR   #^M<R0,R1,R2,R3>        ; Save registers
             50    5A D0 029C           MOVL    R10,R0                  ; Get buffer address
          00000000'EF 16 029F           JSB     EXE$DEANONPAGED         ; Deallocate the pool
                   0F BA 02A5           POPR    #^M<R0,R1,R2,R3>        ; Save registers
                      05 02A7      50$: RSB                             ; Return to caller
                         02A8
             50    0C 3C 02A8      70$: MOVZWL  #SS$_ACCVIO,R0          ; Return error code
                   E9 11 02AB           BRB     40$                     ; Return to caller
                         02AD
```

```
                 2E 50   E9  0265          BLBC    R0,40$                   ; Br if error
                             0268          ;
                             0268          ; Check for counted string, and find actual length if so.
                             0268          ;
                 55  02  D1  0268          CMPL    #CSTRING,R5              ; Is this special string?
                     03  12  026B          BNEQ    10$                      ; Br if not
                 53  84  9A  026D          MOVZBL  (R4)+,R3                 ; Get length and skip length byte
                             0270          ;
                             0270          ; Move the data
                             0270          ;
                     28  BB  0270   10$:   PUSHR   #^M<R3,R5>               ; Save registers
     67   56   00  64  53  2C  0272          MOVC5   R3,(R4),#0,R6,(R7)       ; Move data to user's buffer, zero fill
                     28  BA  0278          POPR    #^M<R3,R5>               ; Restore registers
                     58  D5  027A          TSTL    R8                       ; Did caller want return length?
                     15  13  027C          BEQL    30$                      ; Br if not
                             027E          IFNOWRT #4,(R8),70$              ; Br if longword not writeable
                 56  53  B1  0284          CMPW    R3,R6                    ; See how much was moved
                     07  15  0287          BLEQ    20$                      ; Use valid data length if it fits
                 53  56  B0  0289          MOVW    R6,R3                    ; Else give him "too short" buffer size
              00 53  1F  E2  028C          BBSS    #31,R3,20$               ; And return buffer overflow indicator
                 68  53  D0  0290   20$:   MOVL    R3,(R8)                  ; Return length to user
                 50  01  9A  0293   30$:   MOVZBL  S^#SS$_NORMAL,R0         ; Set success code
                     5A  D5  0296   40$:   TSTL    R10                      ; Any pool deallocation needed?
                     0D  13  0298          BEQL    50$                      ; Br if no
                     0F  BB  029A          PUSHR   #^M<R0,R1,R2,R3>         ; Save registers
                 50  5A  D0  029C          MOVL    R10,R0                   ; Get buffer address
           00000000'EF  16  029F          JSB     EXE$DEANONPAGED          ; Deallocate the pool
                     0F  BA  02A5          POPR    #^M<R0,R1,R2,R3>         ; Save registers
                     05      02A7   50$:   RSB                              ; Return to caller
                             02A8
                 50  0C  3C  02A8   70$:   MOVZWL  #SS$_ACCVIO,R0           ; Return error code
                     E9  11  02AB          BRB     40$                      ; Return to caller
                             02AD
```

```
                  02AD            .SBTTL  SPECIAL - Handle special conditions
                  02AD
                  02AD    ;++
                  02AD    ;
                  02AD    ; FUNCTIONAL DESCRIPTION:
                  02AD    ;
                  02AD    ;       These routines handle data items which must be transformed
                  02AD    ;       before they are returned to the user.  Generally, some
                  02AD    ;       transformation is applied to the data item and the newly
                  02AD    ;       computed item is stored in LOCAL_SPACE on the stack.
                  02AD    ;       The handling routine then changes R4 to point to LOCAL_SPACE
                  02AD    ;       so that MOVEIT will move the item from local storage.
                  02AD    ;
                  02AD    ; CALLING SEQUENCE:
                  02AD    ;
                  02AD    ;       JSB/BSB
                  02AD    ;
                  02AD    ; INPUTS:
                  02AD    ;
                  02AD    ;       R1      item identifier
                  02AD    ;       R3      item length
                  02AD    ;       R4      item address
                  02AD    ;       R6      user buffer length
                  02AD    ;       R9      LKB address
                  02AD    ;       R10     zero
                  02AD    ;
                  02AD    ; IMPLICIT INPUTS:
                  02AD    ;
                  02AD    ;       IPL = IPL$_SYNCH
                  02AD    ;
                  02AD    ; OUTPUTS:
                  02AD    ;
                  02AD    ;       R10     system buffer address to deallocate or zero if none
                  02AD    ;
                  02AD    ; IMPLICIT OUTPUTS:
                  02AD    ;
                  02AD    ;       none
                  02AD    ;
                  02AD    ; ROUTINE VALUE:
                  02AD    ;
                  02AD    ;       SS$_NORMAL      Normal successful completion
                  02AD    ;       SS$_INSFMEM     Insufficient non-paged dynamic memory
                  02AD    ;
                  02AD    ; SIDE EFFECTS:
                  02AD    ;
                  02AD    ;       none
                  02AD    ;--
                  02AD
                  02AD    CHECK_SPC:
                  02AD            ;
                  02AD            ; Registers R7 and R8 are saved at this level and may be used by
                  02AD            ; the action routines without being saved.  Action routines are JSB'ed
                  02AD            ; to with R7 containing the address of LOCAL_SPACE on the stack.
                  02AD            ;
     7E   57   7D 02AD            MOVQ    R7,-(SP)                ; Save registers
     57   0A   D0 02B0            MOVL    #SPECIAL_LEN,R7         ; Get number of table entries
58 FDA5   CF   DE 02B3            MOVAL   SPECIAL,R8              ; Get address of table
```

```
                         02B8
         88   51   B1    02B8    10$:    CMPW    R1,(R8)+                   ; Does entry match item?
              08   13    02BB            BEQL    20$                        ; Yes, go handle it
         58   04   C0    02BD            ADDL    #4,R8                      ; Skip handler address
         F5   57   F5    02C0            SOBGTR  R7,10$                     ; Scan rest of table
              09   11    02C3            BRB     30$                        ; Nothing to do, exit
                         02C5
    57   F8   AD   DE    02C5    20$:    MOVAL   LOCAL_SPACE(FP),R7         ; Load local address for action routine
         50   01   9A    02C9            MOVZBL  S^#SS$_NORMAL,R0           ; Assume success
              98   16    02CC            JSB     @(R8)+                     ; Call action routine
                         02CE
         57   8E   7D    02CE    30$:    MOVQ    (SP)+,R7                   ; Restore registers
                   05    02D1            RSB
                         02D2    ;+
                         02D2    ; Data handling routines
                         02D2    ;-
                         02D2
                         02D2    ;
                         02D2    ;  The PID must be returned as an EPID.
                         02D2    ;  The EPID field of the LKB is valid only on a master copy lock block.
                         02D2    ;
                         02D2    ; Inputs:       R4 -> LKB$L_EPID in LKB
                         02D2    ;               R7 -> Output longword buffer if needed for return
                         02D2    ;               R9 =  Address of LKB
                         02D2    ;
                         02D2
                         02D2    SPC_PID:
              04   E0    02D2            BBS     #LKB$V_MSTCPY,-            ; Br if master copy, R4 is pointing to
         10   2A   A9    02D4                    LKB$W_STATUS(R9),90$       ;  a valid EPID
    50   0C   A9   D0    02D7            MOVL    LKB$L_PID(R9),R0           ; Else, get the IPID
      00000000'EF  16    02DB            JSB     EXE$IPID_TO_EPID           ; Convert to EPID
         67   50   D0    02E1            MOVL    R0,(R7)                    ; Store the EPID
         54   57   D0    02E4            MOVL    R7,R4                      ; Change the item address
         50   01   9A    02E7    90$:    MOVZBL  S^#SS$_NORMAL,R0           ; Return success
                   05    02EA            RSB
                         02EB
                         02EB    ;
                         02EB    ; The lock state is a composite of several fields
                         02EB    ;
                         02EB
                         02EB    SPC_STATE:
                         02EB            ASSUME  LKB$B_GRMODE EQ LKB$B_RQMODE+1
                         02EB            ASSUME  LKB$B_STATE EQ LKB$B_GRMODE+1
         67   84   3C    02EB            MOVZWL  (R4)+,(R7)                 ; Copy modes
    02   A7   64   90    02EE            MOVB    (R4),2(R7)                 ; ..and state
              05   18    02F2            BGEQ    30$                        ; Br if state is okay
02   A7   FF   8F  90    02F4            MOVB    #LKI$C_WAITING,2(R7)       ; Else, map waiting states to same code
         54   57   D0    02F9    30$:    MOVL    R7,R4                      ; Change the item address
                   05    02FC            RSB
                         02FD
                         02FD    ;
                         02FD    ; The lock's parent lock ID must be extracted from another LKB
                         02FD    ;
                         02FD
                         02FD    SPC_PARENT:
              67   D4    02FD            CLRL    (R7)                       ; Assume no PARENT LKB
         54   64   D0    02FF            MOVL    (R4),R4                    ; Get address of PARENT LKB
```

```
                 04   13  0302          BEQL    10$                          ; Br if none
        67    30 A4   D0  0304          MOVL    LKB$L_LKID(R4),(R7)          ; Get LOCKID of owner process
           54    57   D0  0308  10$:    MOVL    R7,R4                        ; Change the item address
                      05  030B          RSB
                          030C
                          030C  ;
                          030C  ; The CSID of master
                          030C  ;
                          030C
                          030C  SPC_SYSTEM:
                 64   D5  030C          TSTL    (R4)                         ; Is CSID zero?
                 10   12  030E          BNEQ    30$                          ; Br if not, CSID is okay
 50     00000000'EF   D0  0310          MOVL    L^CLU$GL_CLUB,R0             ; Get address of cluster block
                 04   13  0317          BEQL    20$                          ; Br if no cluster
        54    60 A0   9E  0319          MOVAB   CLUB$L_LOCAL_CSID(R0),R4 ; Set new item address
           50    01   9A  031D  20$:    MOVZBL  S^#SS$_NORMAL,R0             ; Return success
                      05  0320  30$:    RSB
                          0321
                          0321  ;
                          0321  ; The lock's resource name space is a composite
                          0321  ;
                          0321
                          0321  SPC_NAMSPACE:
                          0321          ASSUME  RSB$B_RMOD EQ RSB$W_GROUP+2
           18    00   EF  0321          EXTZV   #0,#8+16,-                   ; Get the group field and access mode
           67    64       0324                  (R4),(R7)                    ;  3 bytes.
                 64   B5  0326          TSTW    (R4)                         ; Is this group 0? (ie SYSTEM resource)
                 04   12  0328          BNEQ    10$                          ; Br if not, not a system resource
        00 67    1F   E2  032A          BBSS    #LKI$V_SYSNAM,(R7),10$       ; Set the SYSTEM wide indicator
           54    57   D0  032E  10$:    MOVL    R7,R4                        ; Change the item address
                      05  0331          RSB
                          0332
                          0332  ;
                          0332  ; The lock's lock count is the sum of all locks granted on the resource.
                          0332  ;
                          0332
                          0332  SPC_LCKCOUNT:
                 67   D4  0332          CLRL    (R7)                         ; No locks granted yet!
           58    54   D0  0334          MOVL    R4,R8                        ; Copy listhead address
           58    64   D1  0337  10$:    CMPL    (R4),R8                      ; Back at listhead again?
                 07   13  033A          BEQL    20$                          ; Br if yes
                 67   D6  033C          INCL    (R7)                         ; Else, tally one more lock
           54    64   D0  033E          MOVL    (R4),R4                      ; move down list
                 F4   11  0341          BRB     10$                          ; Look for more
           54    57   D0  0343  20$:    MOVL    R7,R4                        ; Change item address
                      05  0346          RSB
                          0347
                          0347  ;
                          0347  ; The remote lock id
                          0347  ;
                          0347
                          0347  SPC_REMLKID:
                 51   DD  0347          PUSHL   R1                           ; Save R1
        51    50 A9   D0  0349          MOVL    LKB$L_RSB(R9),R1             ; Get RSB address
        67    38 A1   D0  034D          MOVL    RSB$L_CSID(R1),(R7)          ; Is the REMLKID valid?
                 03   13  0351          BEQL    10$                          ; Br if not, information is still local
           67    64   D0  0353          MOVL    (R4),(R7)                    ; Else, return real REMLKID
```

```
         54    57  D0 0356   10$:    MOVL    R7,R4                     ; Return item address
               51 8ED0 0359          POPL    R1                        ; Restore R1
                   05 035C          RSB                               ; Return to caller
                      035D
                      035D   ;
                      035D   ; The list of all locks being blocked by this lock.
                      035D   ;
                      035D
                      035D   SPC_BLOCKEDBY:
               06  BB 035D          PUSHR   #^M<R1,R2>                ; Save registers
             0367  30 035F          BSBW    LKI_ALLOCATE              ; Allocate a system buffer
            2A 50  E9 0362          BLBC    R0,50$                    ; Br if resource failure
         58    54  D0 0365          MOVL    R4,R8                     ; Copy RSB wait queue listhead address
         54    52  D0 0368          MOVL    R2,R4                     ; Copy address of system buffer data
               04  E0 036B          BBS     #LKB$V_MSTCPY,-           ; Br if this is the master copy,
         12 2A A9     036D                  LKB$W_STATUS(R9),10$      ;  information is local to this system
      53    50 A9  D0 0370          MOVL    LKB$L_RSB(R9),R3          ; Get RSB address
      53    38 A3  D0 0374          MOVL    RSB$L_CSID(R3),R3         ; Is this a process copy?
               08  13 0378          BEQL    10$                       ; Br if not, information is still local
                      037A          ;
                      037A          ; Lock information is on MASTER system
                      037A          ;
      00000000'GF  16 037A          JSB     G^LKI$SND_BLKBY           ; Send request for all locks BLOCKEDBY
                      0380                                            ;  this lock
               03  11 0380          BRB     30$                       ; Return with status
                      0382          ;
                      0382          ; Lock information is LOCAL to this system
                      0382          ;
             0288  30 0382   10$:   BSBW    LKI$SEARCH_BLOCKEDBY      ; Find all locks BLOCKEDBY this lock
         53    18  B0 0385   30$:   MOVW    #LKI$C_LENGTH,R3          ; Return size of item
   53    53    10  78 0388          ASHL    #16,R3,R3                 ; Move to high word
         53    6A  B0 038C          MOVW    (R10),R3                  ; Get size of returned buffer
               06  BA 038F   50$:   POPR    #^M<R1,R2>                ; Restore registers
               05     0391          RSB
                      0392
                      0392   ;
                      0392   ; The list of all locks blocking this lock.
                      0392   ;
                      0392
                      0392   SPC_BLOCKING:
               06  BB 0392          PUSHR   #^M<R1,R2>                ; Save registers
             0332  30 0394          BSBW    LKI_ALLOCATE              ; Allocate a system buffer
            2A 50  E9 0397          BLBC    R0,50$                    ; Br if resource failure
         58    54  D0 039A          MOVL    R4,R8                     ; Copy RSB wait queue listhead address
         54    52  D0 039D          MOVL    R2,R4                     ; Copy address of system buffer data
               04  E0 03A0          BBS     #LKB$V_MSTCPY,-           ; Br if this is the master copy,
         12 2A A9     03A2                  LKB$W_STATUS(R9),10$      ;  information is local to this system
      53    50 A9  D0 03A5          MOVL    LKB$L_RSB(R9),R3          ; Get RSB address
      53    38 A3  D0 03A9          MOVL    RSB$L_CSID(R3),R3         ; Is this a process copy?
               08  13 03AD          BEQL    10$                       ; Br if not, information is still local
                      03AF          ;
                      03AF          ; Lock information is on MASTER system
                      03AF          ;
      00000000'GF  16 03AF          JSB     G^LKI$SND_BLKING          ; Send request for all locks BLOCKING
                      03B5                                            ;  this lock
               03  11 03B5          BRB     30$                       ; Return with status
                      03B7          ;
```

```
                           03B7          ; Lock information is LOCAL to this system
                           03B7          ;
              01B2   30    03B7  10$:    BSBW    LKI$SEARCH_BLOCKING      ; Find all locks BLOCKING this lock
           53   18   B0    03BA  30$:    MOVW    #LKI$C_LENGTH,R3         ; Return size of item
     53    53   10   78    03BD          ASHL    #16,R3,R3                ; Move to high word
           53   6A   B0    03C1          MOVW    (R10),R3                 ; Get size of returned buffer
                06   BA    03C4  50$:    POPR    #^M<R1,R2>               ; Restore registers
                     05    03C6          RSB
                           03C7
                           03C7  ;
                           03C7  ; The list of all locks associated with the resource.
                           03C7  ;
                           03C7
                           03C7  SPC_LOCKS:
                06   BB    03C7          PUSHR   #^M<R1,R2>               ; Save registers, R3 & R4 are outputs
              02FD   30    03C9          BSBW    LKI_ALLOCATE             ; Allocate a system buffer
           52   50   E9    03CC          BLBC    R0,80$                   ; Br if failure
           58   54   D0    03CF          MOVL    R4,R8                    ; Copy listhead address
           54   52   D0    03D2          MOVL    R2,R4                    ; Set address of return buffer
                04   E0    03D5          BBS     #LKB$V_MSTCPY,-          ; Br if this is the master copy,
          12   2A   A9    03D7                  LKB$W_STATUS(R9),10$     ;  information is local to this system
     53    50   A9   D0    03DA          MOVL    LKB$L_RSB(R9),R3         ; Get RSB address
     53    38   A3   D0    03DE          MOVL    RSB$L_CSID(R3),R3        ; Is this a process copy?
                08   13    03E2          BEQL    10$                      ; Br if not, information is still local
                           03E4          ;
                           03E4          ; Lock information is on MASTER system
                           03E4          ;
     00000000'GF     16    03E4          JSB     G^LKI$SND_LOCKS          ; Send request for all locks associated
                           03EA                                           ;  with this lock
                2B   11    03EA          BRB     70$                      ; Return with status
                           03EC          ;
                           03EC          ; Lock information is LOCAL to this system
                           03EC          ;
           51   56   D0    03EC  10$:    MOVL    R6,R1                    ; Get size of buffer
                           03EF          ASSUME  RSB$L_CVTQFL EQ RSB$L_GRQFL+8
                           03EF          ASSUME  RSB$L_WTQFL EQ RSB$L_CVTQFL+8
           53   03   9A    03EF          MOVZBL  #3,R3                    ; Initialize number of queues to search
           57   58   D0    03F2  30$:    MOVL    R8,R7                    ; Copy listhead address, again
           58   67   D1    03F5  50$:    CMPL    (R7),R8                  ; Back at listhead again?
                14   13    03F8          BEQL    60$                      ; Br if yes
           51   18   C2    03FA          SUBL    #LKI$C_LENGTH,R1         ; Any room left in buffer?
                25   19    03FD          BLSS    90$                      ; Br if not
           57   67   D0    03FF          MOVL    (R7),R7                  ; Else, move down list
     57    C8   A7   9E    0402          MOVAB   -LKB$L_SQFL(R7),R7       ; Point to start of LKB
                23   10    0406          BSBB    LOCK_INFO                ; Get the lock information
     57    38   A7   9E    0408          MOVAB   LKB$L_SQFL(R7),R7        ; Point back to state queue
                E7   11    040C          BRB     50$                      ; Look for more
                           040E  60$:    ASSUME  RSB$L_CVTQFL EQ RSB$L_GRQFL+8
                           040E          ASSUME  RSB$L_WTQFL EQ RSB$L_CVTQFL+8
           58   08   C0    040E          ADDL    #8,R8                    ; Skip to next queue
           DE   53   F5    0411          SOBGTR  R3,30$                   ; Loop if more queues to search
           50   01   9A    0414          MOVZBL  S^#SS$_NORMAL,R0         ; Return success
           53   18   B0    0417  70$:    MOVW    #LKI$C_LENGTH,R3         ; Return size of item
     53    53   10   78    041A          ASHL    #16,R3,R3                ; Move to high word
           53   6A   B0    041E          MOVW    (R10),R3                 ; Get size of returned buffer
                06   BA    0421  80$:    POPR    #^M<R1,R2>               ; Restore registers
                     05    0423          RSB                              ; Return to caller
```

```
                          0424
      50   0601 8F   3C   0424  90$:    MOVZWL  #SS$_BUFFEROVF,R0           ; Return partial success
                F6   11   0429          BRB     80$                         ; Exit
                          042B
                          042B  ;+
                          042B  ; Return Lock Information
                          042B  ;
                          042B  ;       This routine will return the following lock information:
                          042B  ;
                          042B  ;               LKI$_LOCKID     - the lock's lock id
                          042B  ;               LKI$_PID        - the lock's PID
                          042B  ;               LKI$_SYSTEM     - the resource's system id
                          042B  ;               LKI$_STATE      - the locks current state
                          042B  ;               LKI$_REMLKID    - the remote lock id (Process copy LOCKID)
                          042B  ;               LKI$_REMSYSTEM  - the remote system id (Process copy CSID)
                          042B  ;
                          042B  ; Inputs:
                          042B  ;       R2 = Output buffer address
                          042B  ;       R7 = LKB address
                          042B  ;       R10 = Address of beginning of system buffer
                          042B  ;
                          042B  ; Outputs:
                          042B  ;       None
                          042B  ;
                          042B  ; Side Effects:
                          042B  ;       R0 is destroyed
                          042B  ;       (R10) is increased by lock return size
                          042B  ;-
                          042B  LOCK_INFO:
          6A     18  A0   042B          ADDW    #LKI$C_LENGTH,(R10)         ; Tally return size
      82     30 A7   D0   042E          MOVL    LKB$L_LKID(R7),(R2)+        ; Return the LOCKID (MASTER LOCKID)
                          0432          ;
                          0432          ;  The EPID in the LKB is valid only for a master lock block.
                          0432          ;
      50     14 A7   D0   0432          MOVL    LKB$L_EPID(R7),R0           ; Get the EPID
                 04  E0   0436          BBS     #LKB$V_MSTCPY,-             ; Br if master copy lock
          0A  2A A7       0438                  LKB$W_STATUS(R7),10$        ; ...EPID is valid
      50     0C A7   D0   043B          MOVL    LKB$L_PID(R7),R0            ; Get the IPID
    00000000'EF      16   043F          JSB     L^EXE$IPID_TO_EPID          ; Convert to EPID
          82     50  D0   0445  10$:    MOVL    R0,(R2)+                    ; Return the EPID
      50     50 A7   D0   0448          MOVL    LKB$L_RSB(R7),R0            ; Get RSB address
      82     38 A0   D0   044C          MOVL    RSB$L_CSID(R0),(R2)+        ; Return the SYSTEM ID (MASTER CSID)
                 0E  12   0450          BNEQ    30$                         ; Br if okay
  50    00000000'EF  D0   0452          MOVL    L^CLU$GL_CLUB,R0            ; Else, get address of cluster block
                 05  13   0459          BEQL    30$                         ; Br if no cluster
   FC A2     60 A0   D0   045B          MOVL    CLUB$L_LOCAL_CSID(R0),-4(R2) ; Return real CSID
                          0460  30$:    ASSUME  LKB$B_GRMODE EQ LKB$B_RQMODE+1
      82     34 A7   B0   0460          MOVW    LKB$B_RQMODE(R7),(R2)+      ; Copy modes
      82     36 A7   9B   0464          MOVZBW  LKB$B_STATE(R7),(R2)+       ; Copy current state, zero byte
                 05  18   0468          BGEQ    40$                         ; Br if state is okay
   FE A2     FF 8F   90   046A          MOVB    #LKI$C_WAITING,-2(R2)       ; Else, map waiting states to same code
                          046F  40$:    ;
                          046F          ;  The remote CSID and REMLKID are only valid in a master copy
                          046F          ;  lock block.
                          046F          ;
      82     54 A7   D0   046F          MOVL    LKB$L_REMLKID(R7),(R2)+     ; Copy the REMLKID (PROCESS COPY LKID)
      82     58 A7   D0   0473          MOVL    LKB$L_CSID(R7),(R2)+        ; Get the remote CSID (PROCESS_CPY CSID)
```

```
            04  E0  0477             BBS     #LKB$V_MSTCPY,-                 ; Br if master copy
      16 2A A7      0479                     LKB$W_STATUS(R7),90$            ; ...CSID, REMLKID are valid
   F8 A2   30 A7  D0  047C           MOVL    LKB$L_LKID(R7),-8(R2)           ; Else, return the LOCKID as REMLKID
50    00000000'EF  D0  0481          MOVL    L^CLU$GL_CLUB,R0                ; Get the CLUB
            04  13  0488             BEQL    70$                             ; Br if none, return zero CSID
      50   60 A0  D0  048A           MOVL    CLUB$L_LOCAL_CSID(R0),R0; Else, get real CSID
   FC A2   50  D0  048E      70$:    MOVL    R0,-4(R2)                       ; Return real CSID
            05  0492         90$:    RSB
                0493
```

```
                          0493          .SBTTL  GETLKB - Get specified Lock Block
                          0493  ;++
                          0493  ;
                          0493  ; FUNCTIONAL DESCRIPTION:
                          0493  ;
                          0493  ;       Routine to convert a LKID and check privileges. If a valid LKID is
                          0493  ;       specified, the standard conversion routine VERIFYLOCKID is simply
                          0493  ;       called.  If, however, a LKID that implies a "wildcard" LKID (-1 or 0)
                          0493  ;       is specified, then the next active lock is chosen as the LKID to pass
                          0493  ;       to VERIFYLOCKID which then checks the requestor's privilege to obtain
                          0493  ;       information about the lock and returns the lock's LKB address.
                          0493  ;
                          0493  ; CALLING SEQUENCE:
                          0493  ;
                          0493  ;       JSB/BSB
                          0493  ;
                          0493  ; INPUTS:
                          0493  ;
                          0493  ;       R4              current process PCB address
                          0493  ;       LKID(AP)        address of specified LKID
                          0493  ;
                          0493  ; IMPLICIT INPUTS:
                          0493  ;
                          0493  ;       IPL <= IPL$_ASTDEL
                          0493  ;
                          0493  ; OUTPUTS:
                          0493  ;
                          0493  ;       R0      success/failure of operation
                          0493  ;       R4      current process PCB address
                          0493  ;       R9      specified lock's LKB address
                          0493  ;
                          0493  ; COMPLETION CODES:
                          0493  ;
                          0493  ;       SS$_NORMAL      Normal successful completion
                          0493  ;       SS$_ACCVIO      Access violation on attempt to access lock id
                          0493  ;       SS$_NOMORELOCK  No more locks available (on "wildcard" operations)
                          0493  ;
                          0493  ; SIDE EFFECTS:
                          0493  ;
                          0493  ;       R5 and R6 are destroyed.
                          0493  ;--
                          0493
                          0493  GETLKB:
                 55  D4   0493          CLRL    R5                      ; Assume not "wildcard" LKID
         56   08 AC  D0   0495          MOVL    LKID(AP),R6             ; Get LKID address
                 42  13   0499          BEQL    60$                     ; Br if none
                          049B          IFNOWRT #4,(R6),50$             ; Check access to LKID
            51   66  D0   04A1          MOVL    (R6),R1                 ; Get LKID
                 23  14   04A4          BGTR    20$                     ; Br if standard LKID
                          04A6  ;
                          04A6  ; "Wildcard" type LKID specified
                          04A6  ;
            55   51  32   04A6          CVTWL   R1,R5                   ; Get LKIX (LocK IndeX) from LKID
                 02  14   04A9          BGTR    10$                     ; If gtr, valid LKIX
                 55  D4   04AB          CLRL    R5                      ; Else, start with index = 1
                 55  B6   04AD  10$:    INCW    R5                      ; Increment LKIX
   00000000'EF   55  B1   04AF          CMPW    R5,LCK$GL_MAXID         ; Is LKIX in valid range?
```

```
SYSG
Symb

$$T1
ACB$
ACB_
ACB_
ACB_
ACB_
ACB_
ACB
ASTA
ASTR
BSTR
CHEC
CHEC
CLU$
CLUE
CSTR
DYN$
EFN
EXE$
EXE$
EXE$
EXE$
EXE$
EXE_
GETD
GET
GRET
IOC$
IOSB
IPL$
IPL$
ITML
JIB$
JIB$
LCK$
LCK$
LCK$
LCK$
LCK$
LCK$
LCK$
LIM$
LIM$
LIM$
LIM$
LIM$
LKB$
LKB$
LKB$
LKB$
LKB$
LKB$
LKB$
LKB$
LKB$
LKB$
LKB$
```

```
                   25  1A  04B6          BGTRU   60$                        ; Br if not - no more locks
   50   00000000'FF45  D0  04B8          MOVL    @LCK$GL_IDTBL[R5],R0       ; Get LKB address
                   EB  18  04C0          BGEQ    10$                        ; Br if unused slot
        51      30 A0  D0  04C2          MOVL    LKB$L_LKID(R0),R1          ; Get LKID from LKB
             66    51  D0  04C6          MOVL    R1,(R6)                    ; Store LKID in argument list
                           04C9          ;
                           04C9          ; Get LKB and check privileges
                           04C9          ;
                 0018  30  04C9    20$:  BSBW    VERIFYLOCKID               ; Get LKB address and check privileges
                   55  B5  04CC          TSTW    R5                         ; "wildcard" type LKID specified?
                   07  13  04CE          BEQL    40$                        ; Br if not
                DA 50  E9  04D0          BLBC    R0,10$                     ; Br if error, return only "good" ones
           02 A6   01  AE  04D3          MNEGW   #1,2(R6)                   ; Else, set continuation context
                       05  04D7    40$:  RSB                                ; Return to caller
                           04D8
             50    0C  3C  04D8    50$:  MOVZWL  #SS$_ACCVIO,R0             ; Set access violation
                   FA  11  04DB          BRB     40$                        ;
   50        0A08  8F  3C  04DD    60$:  MOVZWL  #SS$_NOMORELOCK,R0         ; Set no more processes
                   F3  11  04E2          BRB     40$                        ;
                           04E4
```

```
SYSG
Symb

PSL$
PSL$
RESE
RSB$
RSB$
RSB$
RSB$
RSB$
RSB$
RSB$
RSB$
RSB$
RSBT
SAVE
SCH$
SCH$
SCH$
SPC_
SPC_
SPC_
SPC_
SPC_
SPC_
SPC_
SPC_
SPC_
SPC_
SPEC
SPEC
SS$_
SS$_
SS$_
SS$_
SS$_
SS$_
SS$_
SS$_
SS$_
SS$_
SS$_
SYS$
VALU
VERI

PSEC
----
.  A
$ABS
WSYS
YEXE
```

```
                               04E4         .SBTTL  VERIFYLOCKID - Verify lock id
                               04E4
                               04E4   ;++
                               04E4   ; FUNCTIONAL DESCRIPTION:
                               04E4   ;
                               04E4   ;       This routine verifies a lock id for correct process ownership
                               04E4   ;       and access mode and then converts it into a LKB address.
                               04E4   ;
                               04E4   ;       LKB is not locked after leaving this routine, therefore we
                               04E4   ;       must re-verify the LKB everytime we attempt to use it.
                               04E4   ;
                               04E4   ; CALLING SEQUENCE:
                               04E4   ;
                               04E4   ;       JSB/BSB
                               04E4   ;
                               04E4   ;       Note:  IPL is raised to IPL$_SYNCH to prevent the owner of
                               04E4   ;       the lock from releasing the [KB/RSB in the middle of verifying
                               04E4   ;       its lock id.
                               04E4   ;
                               04E4   ; INPUTS:
                               04E4   ;
                               04E4   ;       R1      Lock id
                               04E4   ;       R4      Address of PCB
                               04E4   ;       R5      Zero if not a wildcard search operation
                               04E4   ;
                               04E4   ; OUTPUTS:
                               04E4   ;
                               04E4   ;       R0      Completion code
                               04E4   ;       R9      Address of LKB
                               04E4   ;
                               04E4   ; COMPLETION CODES:
                               04E4   ;
                               04E4   ;       SS$_NORMAL      Lock id was valid and converted to LKB address
                               04E4   ;       SS$_IVLOCKID    Invalid lock id
                               04E4   ;       SS$_IVMODE      Access mode violation on attempt to access lock
                               04E4   ;       SS$_NOSYSLCK    No SYSLCK privilege to access system lock
                               04E4   ;       SS$_NOWORLD     No WORLD privilege to access lock
                               04E4   ;
                               04E4   ; SIDE EFFECTS:
                               04E4   ;
                               04E4   ;       R0 and R1 are destroyed
                               04E4   ;--
                               04E4
                               04E4           ASSUME  LKB$V_MODE  EQ  0
                               04E4           ASSUME  LKB$S_MODE  EQ  2
                               04E4
                               04E4   VERIFYLOCKID:
                               04E4           DSBINT  #IPL$_SYNCH             ; Raise IPL to sync access to LKBs
              59     51    3C  04EA           MOVZWL  R1,R9                   ; Put lockid index in R9
      00000000'EF    59    D1  04ED           CMPL    R9,LCK$GL_MAXID         ; Is the lock id too big?
                     5A    1A  04F4           BGTRU   40$                     ; Yes
  59  00000000'FF49        D0  04F6           MOVL    @LCK$GL_IDTBL[R9],R9    ; Get LKB address
                     50    18  04FE           BGEQ    40$                     ; Unallocated id
           30 A9     51    D1  0500           CMPL    R1,LKB$L_LKID(R9)       ; Check sequence number
                     4A    12  0504           BNEQ    40$                     ; Not valid
        50    50 A9        D0  0506           MOVL    LKB$L_RSB(R9),R0        ; Get RSB address
              4C A0        B5  050A           TSTW    RSB$W_GROUP(R0)         ; Is this a system resource?
```

```
SYSG
VAX-

Phas
----
Init
Comm
Pass
Symb
Pass
Symb
Psec
Cros
Asse

The
1138
Ther
1661
39 p

Macr
----
_$25
_$25
_$25
TOTA

1957

Ther

MACR
```

```
                   17   13  050D         BEQL     10$                       ; Br if yes
   51     00000000'GF   D0  050F         MOVL     G^SCH$GL_CURPCB,R1        ; Else, get our PCB address
               00BE C1  B1  0516         CMPW     PCB$W_GRP(R1),-           ; Do we have group access to LKB?
                 4C A0      051A                  RSB$W_GROUP(R0)           ; ..no privilege needed
                   1A   13  051C         BEQL     20$                       ; Br if our group - always allowed
                            051E         IFNPRIV  WORLD,70$                 ; Br if NO privilege to access lock
                   12   11  0524         BRB      20$                       ; Else, success
                   50   DC  0526 10$:    MOVPSL   R0                        ; Get current PSL
                   16   EF  0528         EXTZV    #PSL$V_PRVMOD,-           ; Extract previous mode field
          50    50 02       052A                  #PSL$S_PRVMOD,R0,R0
                            052D         ASSUME   PSL$C_KERNEL EQ 0
                            052D         ASSUME   PSL$C_EXEC EQ 1
               50  01   91  052D         CMPB     #PSL$C_EXEC,R0            ; Does the user have the right access
                            0530                                            ;  mode to access the LKB?
                   06   1E  0530         BGEQU    20$                       ; Br if yes
                            0532         IFNPRIV  SYSLCK,60$                ; Br if NO privilege to look at lock
                   50   DC  0538 20$:    MOVPSL   R0                        ; Get current PSL
                   16   EF  053A         EXTZV    #PSL$V_PRVMOD,-           ; Extract previous mode field
          50    50 02       053C                  #PSL$S_PRVMOD,R0,R0
           51    50 A9  D0  053F         MOVL     LKB$L_RSB(R9),R1          ; Get RSB address
           4E A1    50  91  0543         CMPB     R0,RSB$B_RMOD(R1)         ; Caller have privilege to access lock?
                   0E   1A  0547         BGTRU    50$                       ; Br if No
               50  01   9A  0549         MOVZBL   S^#SS$_NORMAL,R0          ; Else, Yes - return success
                            054C 30$:    ENBINT                             ; Restore IPL
                        05  054F         RSB
                            0550
          50   2124 8F  3C  0550 40$:    MOVZWL   #SS$_IVLOCKID,R0          ; Invalid lock id
                   F5   11  0555         BRB      30$                       ; Leave
          50   0354 8F  3C  0557 50$:    MOVZWL   #SS$_IVMODE,R0            ; Illegal access mode
                   EE   11  055C         BRB      30$                       ; Leave
          50   28F4 8F  3C  055E 60$:    MOVZWL   #SS$_NOSYSLCK,R0          ; No SYSLCK privilege to access lock
                   E7   11  0563         BRB      30$                       ; Leave
          50   2884 8F  3C  0565 70$:    MOVZWL   #SS$_NOWORLD,R0           ; No WORLD privilege to access lock
                   E0   11  056A         BRB      30$                       ; Leave
                            056C
                   056C  1263          .SBTTL  LKI$SEARCH_BLOCKING - Search for locks blocking the current lock
                   056C  1264
                   056C  1265  ;++
                   056C  1266  ; FUNCTIONAL DESCRIPTION:
                   056C  1267  ;
                   056C  1268  ;       This routine searches for locks blocking the current lock.  A
                   056C  1269  ;       blocking lock is one in which the maximized request mode is
                   056C  1270  ;       incompatible with the requested mode (if the lock is on the
                   056C  1271  ;       waiting or conversion queue) or the granted mode (if the lock
                   056C  1272  ;       is on the granted queue).
                   056C  1273  ;
                   056C  1274  ;       For example, assume there is PR locks granted on a resource and
                   056C  1275  ;       a second user issues an EX mode request on the resource. The first
                   056C  1276  ;       lock is now BLOCKING the second lock and the first lock would be
                   056C  1277  ;       returned in list of locks BLOCKING the second lock.
                   056C  1278  ;
                   056C  1279  ;       To find BLOCKING locks it is sufficient to check all locks
                   056C  1280  ;       ahead of this lock on all queues (in th order, REQESTED,
                   056C  1281  ;       CONVERSION and then GRANTED) to see if their requested or granted
                   056C  1282  ;       modes are incompatible with this locks requested mode.
                   056C  1283  ;
                   056C  1284  ; CALLING SEQUENCE:
                   056C  1285  ;
                   056C  1286  ;       JSB/BSB
                   056C  1287  ;
                   056C  1288  ; INPUTS:
                   056C  1289  ;
                   056C  1290  ;       R2      address of system buffer for storing the lock information
                   056C  1291  ;       R6      length of system buffer for storing the lock information
                   056C  1292  ;       R8      address of wait queue in RSB
                   056C  1293  ;       R9      LKB address
                   056C  1294  ;
                   056C  1295  ; IMPLICIT INPUTS:
                   056C  1296  ;
                   056C  1297  ;       IPL = IPL$_SYNCH
                   056C  1298  ;
                   056C  1299  ; OUTPUTS:
                   056C  1300  ;
                   056C  1301  ;       R0      always success!
                   056C  1302  ;
                   056C  1303  ; SIDE EFFECTS:
                   056C  1304  ;
                   056C  1305  ;       R7 is destroyed.
                   056C  1306  ;--
                   056C  1307
                   056C  1308  LKI$SEARCH_BLOCKING::
 0066 8F     BB    056C  1309          PUSHR   #^M<R1,R2,R5,R6>         ; Save registers
                   0570  1310          ;
                   0570  1311          ; First run through all locks waiting ahead of this lock
                   0570  1312          ; maximizing the requested modes and checking all locks
                   0570  1313          ; incompatible with the current "maxmode".  If this lock is
                   0570  1314          ; on the wait queue then we do the wait queue first and
                   0570  1315          ; the converison queue next.  If this lock is on the
                   0570  1316          ; conversion queue then we do only the conversion queue.
                   0570  1317          ; Later we'll do all the granted locks.
                   0570  1318          ;
                   0570  1319          ; If this lock is on the granted queue, we skip right to the
```

```
                            0570              ; search of the granted queue locks.
                            0570              ;
                            0570              ASSUME  LKB$K_GRANTED  EQ  1
                            0570              ASSUME  LKB$K_CONVERT  EQ  0
                            0570              ASSUME  LKB$K_WAITING  EQ  -1
                            0570              ASSUME  RSB$L_CVTQFL   EQ  RSB$L_GRQFL+8
                            0570              ASSUME  RSB$L_WTQFL    EQ  RSB$L_CVTQFL+8
                            0570
        55   34 A9    9A    0570              MOVZBL  LKB$B_RQMODE(R9),R5      ; Get the current lock's requested mode
             57   59    D0    0574              MOVL    R9,R7                    ; R7 will point to other LKB's
                            0577                                               ; in front of the one pointed to by R9
             36 A9    95    0577              TSTB    LKB$B_STATE(R9)          ; Which queue is lock on?
                63    14    057A              BGTR    60$                      ; Br if granted queue
                03    19    057C              BLSS    10$                      ; Br if waiting queue
                            057E              ;
                            057E              ;    Lock is on the conversion queue
                            057E              ;
             58   08    C2    057E              SUBL    #8,R8                    ; Point to conversion queue header
                            0581
        57   3C A7    D0    0581         10$: MOVL    LKB$L_SQBL(R7),R7        ; Get previous lock on state queue
             58   57    D1    0585              CMPL    R7,R8                    ; Reached head of queue yet?
                42    13    0588              BEQL    50$                      ; Br if yes
             57   38    C2    058A              SUBL    #LKB$L_SQFL,R7           ; Back up to point at start of LKB
        50   34 A7    9A    058D              MOVZBL  LKB$B_RQMODE(R7),R0      ; R0 = requested mode
             51   55    D0    0591              MOVL    R5,R1                    ; Save old maxmode
                            0594              ;
                            0594              ; Maximize lock modes (in R0 and R5) and see if this lock (R7) is
                            0594              ; incompatible with (the previous) maxmode.  The maximization function
                            0594              ; is a simple arithmetic maximum except if the two modes are CW and PR.
                            0594              ; In that case the maximum of CW and PR is PW.  PW is incompatible
                            0594              ; with everything either CW or PR is incompatible with.
                            0594              ;
             55   50    91    0594              CMPB    R0,R5                    ; Current mode greater than maxmode?
                20    13    0597              BEQL    35$                      ; Br if No, they're equal
                0C    1A    0599              BGTRU   20$                      ; Br if Yes, compute new maxmode
             02   50    91    059B              CMPB    R0,#LCK$K_CWMODE         ; Br if No, is current mode CW?
                19    12    059E              BNEQ    35$                      ; Br if No, maxmode = R2
             03   55    91    05A0              CMPB    R5,#LCK$K_PRMODE         ; Br if Yes, is maxmode PR?
                14    12    05A3              BNEQ    35$                      ; Br if No, maxmode = R2
                0A    11    05A5              BRB     25$                      ; Br if Yes, new maxmode is PW
             02   55    91    05A7         20$: CMPB    R5,#LCK$K_CWMODE         ; Is maxmode CW?
                0A    12    05AA              BNEQ    30$                      ; Br if No, maxmode = R0
             03   50    91    05AC              CMPB    R0,#LCK$K_PRMODE         ; Br if Yes, is current mode PR?
                05    12    05AF              BNEQ    30$                      ; Br if No, maxmode = R0
             55   04    90    05B1         25$: MOVB    #LCK$K_PWMODE,R5         ; Have CW and PR; maxmode = PW
                03    11    05B4              BRB     35$
             55   50    90    05B6         30$: MOVB    R0,R5                    ; Maxmode = R0
                            05B9
00000000'EF41     50    E0    05B9         35$: BBS     R0,-                     ; Branch if compatible with
                BF          05C1                      L^LCK$COMPAT_TBL[R1],10$;  saved maxmode
                            05C2              ;
                            05C2              ; Have a lock incompatible with maxmode, return the lock info.
                            05C2              ;
             56   18    C2    05C2              SUBL    #LKI$C_LENGTH,R6         ; Any room left in buffer?
                3E    19    05C5              BLSS    90$                      ; Br if not, leave now
              FE61    30    05C7              BSBW    LOCK_INFO                ; Return the lock information
                B5    11    05CA         40$: BRB     10$                      ; Get next lock in RSB (outer loop)
```

```
                                  05CC
                                  05CC  50$:    ;
                                  05CC          ; Reached the queue header.  Back up R8 to point to the previous
                                  05CC          ; queue header in the RSB.  If R8 is pointing to the granted
                                  05CC          ; queue, then we are done with this loop and we continue with
                                  05CC          ; the granted queue.  Otherwise, we repeat this loop for the
                                  05CC          ; conversion queue.
                                  05CC          ;
                 58    08    C2   05CC          SUBL    #8,R8                   ; Back up R8 one queue header
              57    C8 A8    9E   05CF          MOVAB   -LKB$L_SQFL(R8),R7      ; Prepare to process that queue
                       10    C1   05D3          ADDL3   #RSB$L_GRQFL,-          ; Get address of granted queue
              50    50 A9         05D5                  LKB$L_RSB(R9),R0
                 50    58    D1   05D8          CMPL    R8,R0                   ; Have we reached the granted queue?
                       ED    12   05DB          BNEQ    40$                     ; Br if Not, repeat for conversion queue
                                  05DD          ;
                                  05DD          ; Now repeat a similar procedure for all locks on the granted
                                  05DD          ; queue whose granted mode is incompatible with the maxmode
                                  05DD          ; in R5.
                                  05DD          ;
                       03    11   05DD          BRB     70$
                                  05DF
                                  05DF  60$:    ;
                                  05DF          ;  Lock is initially on the granted queue.
                                  05DF          ;
                 58    10    C2   05DF          SUBL    #16,R8                  ; Point to granted queue header
                                  05E2
              57    3C A7    D0   05E2  70$:    MOVL    LKB$L_SQBL(R7),R7       ; Get next lock in granted queue
                 58    57    D1   05E6          CMPL    R7,R8                   ; Reached end of queue?
                       1A    13   05E9          BEQL    90$                     ; Br if Yes, all done
                 57    38    C2   05EB          SUBL    #LKB$L_SQFL,R7          ; Back up to point at start of LKB
              50    35 A7    9A   05EE          MOVZBL  LKB$B_GRMODE(R7),R0     ; Get granted mode
 E7 00000000'EF45      50    E0   05F2          BBS     R0,L^[CK$COMPAT_TBL[R5],70$ ; Branch if compatible
                                  05FB          ;
                                  05FB          ; Have an incompatible lock on the granted queue, return lock info.
                                  05FB          ;
                 56    18    C2   05FB          SUBL    #LKI$C_LENGTH,R6        ; Any room left in buffer?
                       05    19   05FE          BLSS    90$                     ; Br if not, leave now
                     FE28    30   0600          BSBW    LOCK_INFO               ; Return lock info
                       DD    11   0603          BRB     70$                     ; Look for more
                                  0605
                 50    01    9A   0605  90$:    MOVZBL  #1,R0                   ; Success indicator
               0066 8F       BA   0608          POPR    #^M<R1,R2,R5,R6>        ; Restore registers
                             05   060C          RSB
                                  060D
                 060D         .SBTTL  LKI$SEARCH_BLOCKEDBY - Search for locks blockedby the current lock
                 060D
                 060D ;++
                 060D ; FUNCTIONAL DESCRIPTION:
                 060D ;
                 060D ;        This routine searches for locks blocked by the current lock.
                 060D ;        A blocked lock is one which is either blocked by the current
                 060D ;        lock or is blocked by any other lock blocked by the current
                 060D ;        lock. We must start with the current lock on whatever queue
                 060D ;        it may currently be on and then maximize the requested for
                 060D ;        locks on the converting or waiting queues. All locks are checked
                 060D ;        to see if the maximized request mode is incompatible with the
                 060D ;        requested mode (if the locks is not on the granted queue).
                 060D ;
                 060D ;        For example, assume there is an EX lock granted on a resource and
                 060D ;        a two other users have issued PR requests on the resource. Now
                 060D ;        if we wish to find all locks BLOCKEDBY the first lock, then the
                 060D ;        list consists of the two locks waiting for the resource in PR
                 060D ;        mode.
                 060D ;
                 060D ;        To find BLOCKING locks it is sufficient to check all locks
                 060D ;        behing the current lock on all queues (in the order, GRANTED
                 060D ;        CONVERTING and then WAITING) to see if their requested mode
                 060D ;        is incompatible with the current lock's requested (or granted)
                 060D ;        mode. Once, we have found one blocked lock, then that lock and all
                 060D ;        locks following are also blocked.
                 060D ;
                 060D ; CALLING SEQUENCE:
                 060D ;
                 060D ;        JSB/BSB
                 060D ;
                 060D ; INPUTS:
                 060D ;
                 060D ;        R2      address of system buffer for storing the lock information
                 060D ;        R6      length of system buffer for storing the lock information
                 060D ;        R8      address of wait queue in RSB
                 060D ;        R9      LKB address
                 060D ;
                 060D ; IMPLICIT INPUTS:
                 060D ;
                 060D ;        IPL = IPL$_SYNCH
                 060D ;
                 060D ; OUTPUTS:
                 060D ;
                 060D ;        R0      always success!
                 060D ;
                 060D ; SIDE EFFECTS:
                 060D ;
                 060D ;        R7 is destroyed.
                 060D ;--
                 060D
                 060D LKI$SEARCH_BLOCKEDBY::
0066 8F    BB    060D         PUSHR   #^M<R1,R2,R5,R6>          ; Save registers
                 0611         ;
                 0611         ; First run through all locks waiting behind this lock
                 0611         ; maximizing the requested modes and checking all locks
                 0611         ; incompatible with the current "maxmode".  If we find a
```

```
                          0611              ; lock that is blocked by the current lock, then that lock
                          0611              ; and all the following locks are blocked. For locks that
                          0611              ; are on the granted queue we do not maximize the granted
                          0611              ; mode, for all other queues we will maximize the request mode.
                          0611              ;
                          0611              ; If this lock is not on the granted queue, we skip right to the
                          0611              ; search of the other queue locks.
                          0611              ;
                          0611              ASSUME  LKB$K_GRANTED   EQ  1
                          0611              ASSUME  LKB$K_CONVERT   EQ  0
                          0611              ASSUME  LKB$K_WAITING   EQ  -1
                          0611              ASSUME  RSB$L_CVTQFL  EQ  RSB$L_GRQFL+8
                          0611              ASSUME  RSB$L_WTQFL   EQ  RSB$L_CVTQFL+8
                          0611
         55    34 A9   9A 0611              MOVZBL  LKB$B_RQMODE(R9),R5     ; Get the current lock's requested mode
            57    59   D0 0615              MOVL    R9,R7                   ; R7 will point to other LKB's
                          0618                                              ;  after the one pointed to by R9
               36 A9   95 0618              TSTB    LKB$B_STATE(R9)         ; Which queue is lock on?
                  21   19 061B              BLSS    20$                     ; Br if waiting
                  22   13 061D              BEQL    30$                     ; Br if converting
                          061F              ;
                          061F              ;   Lock is on the granted queue
                          061F              ;
         55    35 A9   9A 061F              MOVZBL  LKB$B_GRMODE(R9),R5     ; Get the current lock's granted mode
                          0623
         57    38 A7   D0 0623        10$:  MOVL    LKB$L_SQFL(R7),R7       ; Get next lock on state queue
            58    57   D1 0627              CMPL    R7,R8                   ; Reached head of queue yet?
                  5B   13 062A              BEQL    90$                     ; Br if yes
            57    38   C2 062C              SUBL    #LKB$L_SQFL,R7          ; Back up to point at start of LKB
         50    35 A7   9A 062F              MOVZBL  LKB$B_GRMODE(R7),R0     ; Get the lock's granted mode
00000000'EF45     50   E0 0633              BBS     R0,-                    ; Branch if compatible
                  E7      063B              L^LCK$COMPAT_TBL[R5],10$;
                          063C              ;
                          063C              ; Have an incompatible, return the lock info. for all succeeding locks
                          063C              ;
                  62   11 063C              BRB     120$                    ; Return lock info.
                          063E
                          063E        20$:  ;
                          063E              ; Lock is initially on the waiting queue.
                          063E              ;
            58    08   C0 063E              ADDL    #8,R8                   ; Advance R8 one queue header
                          0641        30$:  ;
                          0641              ; Lock is initially on the converting queue, OR we have
                          0641              ; reached the queue header. Advance R8 to point to the next
                          0641              ; queue header in the RSB.
                          0641              ;
            58    08   C0 0641              ADDL    #8,R8                   ; Advance R8 one queue header
                          0644              ;
                          0644              ; Run thru all locks on either the converting or waiting queue
                          0644              ; lock for any locks blocked by the maxmode in R5.
                          0644              ;
         57    38 A7   D0 0644        40$:  MOVL    LKB$L_SQFL(R7),R7       ; Get next lock in queue
            58    57   D1 0648              CMPL    R7,R8                   ; Reached end of queue?
                  3A   13 064B              BEQL    90$                     ; Br if Yes, all done
            57    38   C2 064D              SUBL    #LKB$L_SQFL,R7          ; Back up to point at start of LKB
         50    34 A7   9A 0650              MOVZBL  LKB$B_RQMODE(R7),R0     ; Get requested mode
            51    55   D0 0654              MOVL    R5,R1                   ; Save old maxmode
```

```
                         0657           ;
                         0657           ; Maximize lock modes (in R0 and R5) and see if this lock (R7) is
                         0657           ; incompatible with (the previous) maxmode.  The maximization function
                         0657           ; is a simple arithmetic maximum except if the two modes are CW and PR.
                         0657           ; In that case the maximum of CW and PR is PW.  PW is incompatible
                         0657           ; with everything either CW or PR is incompatible with.
                         0657           ;
            55    50  91 0657           CMPB    R0,R5                     ; Current mode greater than maxmode?
                  20  13 065A           BEQL    80$                       ; Br if No, they're equal
                  0C  1A 065C           BGTRU   50$                       ; Br if Yes, compute new maxmode
            02    50  91 065E           CMPB    R0,#LCK$K_CWMODE          ; Br if No, is current mode CW?
                  19  12 0661           BNEQ    80$                       ; Br if No, maxmode = R2
            03    55  91 0663           CMPB    R5,#LCK$K_PRMODE          ; Br if Yes, is maxmode PR?
                  14  12 0666           BNEQ    80$                       ; Br if No, maxmode = R2
                  0A  11 0668           BRB     60$                       ; Br if Yes, new maxmode is PW
            02    55  91 066A   50$:    CMPB    R5,#LCK$K_CWMODE          ; Is maxmode CW?
                  0A  12 066D           BNEQ    70$                       ; Br if No, maxmode = R0
            03    50  91 066F           CMPB    R0,#LCK$K_PRMODE          ; Br if Yes, is current mode PR?
                  05  12 0672           BNEQ    70$                       ; Br if No, maxmode = R0
            55    04  90 0674   60$:    MOVB    #LCK$K_PWMODE,R5          ; Have CW and PR; maxmode = PW
                  03  11 0677           BRB     80$
            55    50  90 0679   70$:    MOVB    R0,R5                     ; Maxmode = R0
                         067C
   00000000'EF41  50  E1 067C   80$:    BBC     R0,-                      ; Branch if incompatible
                  1B     0684                   L^LCK$COMPAT_TBL[R1],120$;  with saved maxmode
                  BD  11 0685           BRB     40$                       ; Else, check next lock in RSB
                         0687
            58    08  C0 0687   90$:    ADDL    #8,R8                     ; Advance R8 one queue header
         57    C8 A8  9E 068A           MOVAB   -LKB$L_SQFL(R8),R7        ; Prepare to process that queue
                  28  C1 068E           ADDL3   #RSB$L_WTQFL+8,-          ; Get address past waiting queue
      50    50 A9        0690                   LKB$L_RSB(R9),R0
            50    58  D1 0693           CMPL    R8,R0                     ; Have we done all the queues?
                  AC  12 0696           BNEQ    40$                       ; Br if Not, repeat for remaining queue
                         0698
            50    01  9A 0698   100$:   MOVZBL  #1,R0                     ; Success indicator
          0066 8F     BA 069B           POPR    #^M<R1,R2,R5,R6>          ; Restore registers
                      05 069F           RSB
                         06A0
                         06A0           ;
                         06A0           ; We have found the first incompatible lock
                         06A0           ;
            56    18  C2 06A0   120$:   SUBL    #LKI$C_LENGTH,R6          ; Any room left in buffer?
                  F3  19 06A3           BLSS    100$                      ; Br if not
                FD83  30 06A5           BSBW    LOCK_INFO                 ; Else, return lock info.
         57    38 A7  D0 06A8   130$:   MOVL    LKB$L_SQFL(R7),R7         ; Get next lock in queue
            58    57  D1 06AC           CMPL    R7,R8                     ; Reached end of queue?
                  05  13 06AF           BEQL    140$                      ; Br if Yes, skip to next queue
            57    38  C2 06B1           SUBL    #LKB$L_SQFL,R7            ; Back up to point at start of LKB
                  EA  11 06B4           BRB     120$                      ; Return the lock info.
                         06B6
            58    08  C0 06B6   140$:   ADDL    #8,R8                     ; Advance R8 one queue header
         57    C8 A8  9E 06B9           MOVAB   -LKB$L_SQFL(R8),R7        ; Prepare to process that queue
                  28  C1 06BD           ADDL3   #RSB$L_WTQFL+8,-          ; Get address past end of queues
      50    50 A9        06BF                   LKB$L_RSB(R9),R0
            50    58  D1 06C2           CMPL    R8,R0                     ; Have we done all queues?
                  D1  13 06C5           BEQL    100$                      ; Br if Yes, leave
                  DF  11 06C7           BRB     130$                      ; Else, loop thru remaining queues
```

```
                               06C9  1593
```

```
                             06C9       .SBTTL  LKI_ALLOCATE - Allocate a system buffer
                             06C9
                             06C9  ;++
                             06C9  ; FUNCTIONAL DESCRIPTION:
                             06C9  ;
                             06C9  ;        This routine attempts to allocate a system buffer and intialize
                             06C9  ;        the structure type.
                             06C9  ;
                             06C9  ; CALLING SEQUENCE:
                             06C9  ;
                             06C9  ;        JSB/BSB
                             06C9  ;
                             06C9  ; INPUTS:
                             06C9  ;
                             06C9  ;        R6      Size of desired buffer minus header
                             06C9  ;
                             06C9  ; IMPLICIT INPUTS:
                             06C9  ;
                             06C9  ;        IPL = IPL$_SYNCH
                             06C9  ;
                             06C9  ; OUTPUTS:
                             06C9  ;
                             06C9  ;        R0      Completion status for request
                             06C9  ;        R2      Address of the system buffer at data portion of buffer
                             06C9  ;        R10     Address of start of the system buffer
                             06C9  ;
                             06C9  ; SIDE EFFECTS:
                             06C9  ;
                             06C9  ;        none
                             06C9  ;--
                             06C9
                             06C9  LKI_ALLOCATE:
                 1A     BB   06C9           PUSHR   #^M<R1,R3,R4>           ; Save registers
  54   00000000'EF     D0   06CB           MOVL    SCH$GL_CURPCB,R4        ; Get PCB address
       51    56  0C     C1   06D2           ADDL3   #12,R6,R1               ; Compute size of system buffer
                             06D6           ;
                             06D6           ; NOTE: The exec routine EXE$BUFFRQUOTA cannot be called, since
                             06D6           ; it will lower IPL and destroy all synchronization.
                             06D6           ;
  50   00000000'EF     3C   06D6           MOVZWL  IOC$GW_MAXBUF,R0        ; Get maximum buffer size allowed
            50   51     D1   06DD           CMPL    R1,R0                   ; Is buffer too big?
                 28     1A   06E0           BGTRU   20$                     ; Br if yes, error
     50   0080 C4     D0   06E2           MOVL    PCB$L_JIB(R4),R0        ; Get JIB address
        24 A0    51     D1   06E7           CMPL    R1,JIB$L_BYTLM(R0)      ; Is BYTLM quota okay?
                 1D     1A   06EB           BGTRU   20$                     ; Br if not, error
        20 A0    51     D1   06ED           CMPL    R1,JIB$L_BYTCNT(R0)     ; Is BYTCNT quota okay?
                 17     1A   06F1           BGTRU   20$                     ; Br if not, error
       00000000'EF     16   06F3           JSB     EXE$ALONONPAGED         ; Try and allocate a buffer
              13 50     E9   06F9           BLBC    R0,30$                  ; Br if failed
            5A   52     D0   06FC           MOVL    R2,R10                  ; Set address of buffer to deallocate
                             06FF           ;
                             06FF           ; Initialize structure header
                             06FF           ;
                 82     7C   06FF           CLRQ    (R2)+                   ; Zero return size, unused fields
            82   51     B0   0701           MOVW    R1,(R2)+                ; Set structure size
            82   13     B0   0704           MOVW    #DYN$C_BUFIO,(R2)+      ; Set structure type
                 1A     BA   0707  10$:     POPR    #^M<R1,R3,R4>           ; Restore registers
```

```
                05  0709          RSB
                    070A
      50   1C   3C  070A  20$:    MOVZWL  #SS$_EXQUOTA,R0         ; Set error return
           F8   11  070D          BRB     10$                     ; Return to caller
                    070F
   50 0124 8F   3C  070F  30$:    MOVZWL  #SS$_INSFMEM,R0         ; Set error return
           F1   11  0714          BRB     10$                     ; Return to caller
                    0716
                    0716
                    0716          .END
```

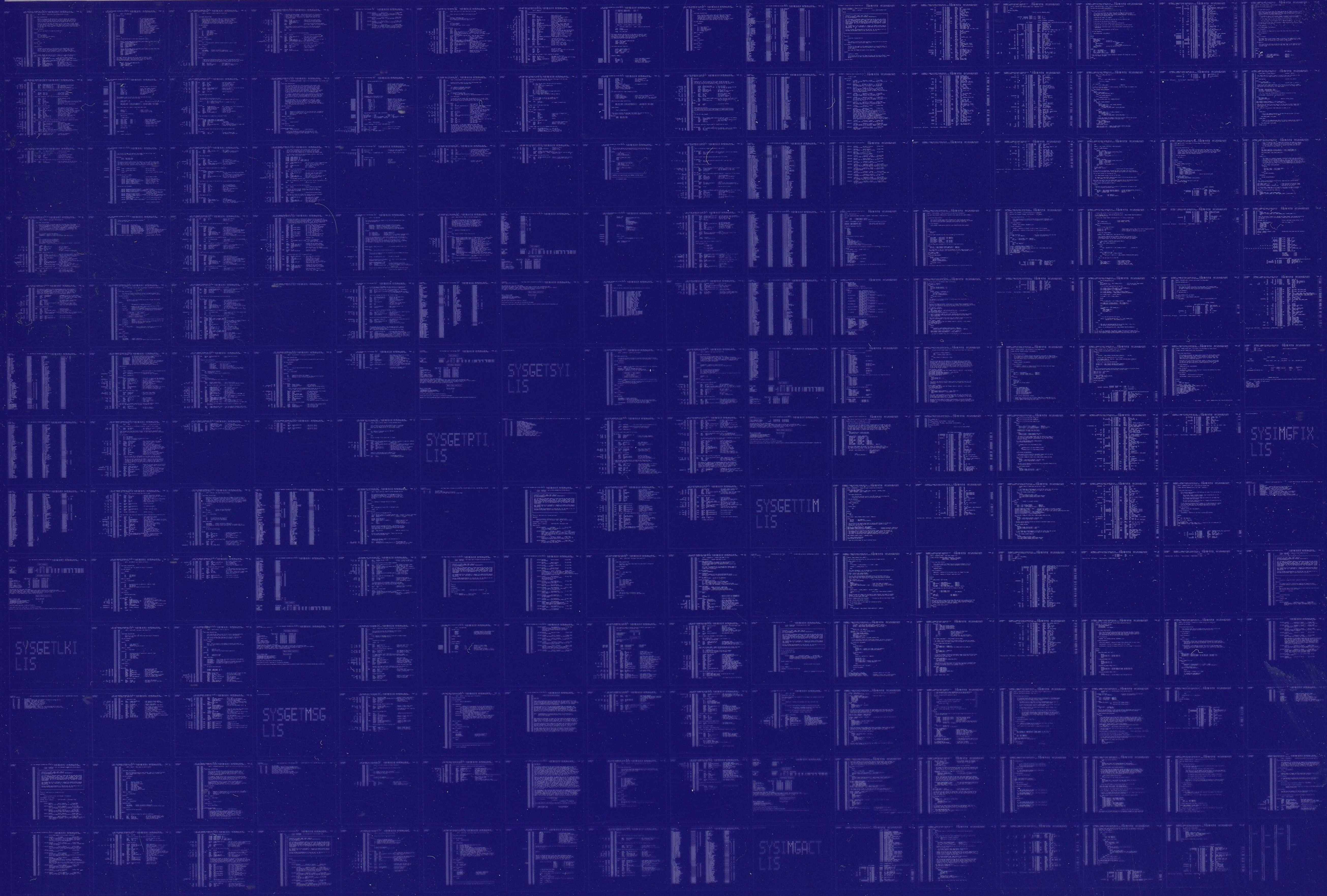
```
$$T1                           = 00000000             LKB$L_REMLKID                  = 00000054
ACB$L_KAST                     = 00000018             LKB$L_RSB                      = 00000050
ACB_L_COUNT                      0000002C             LKB$L_SQBL                     = 0000003C
ACB_L_DADDR                      0000001C             LKB$L_SQFL                     = 00000038
ACB_L_EFN                        00000020             LKB$S_MODE                     = 00000002
ACB_L_ILIST                      00000030             LKB$V_MODE                     = 00000000
ACB_L_IOSB                       00000024             LKB$V_MSTCPY                   = 00000004
ACB_L_OPID                       00000028             LKB$W_REFCNT                   = 0000004C
ASTADR                         = 00000014             LKB$W_STATUS                   = 0000002A
ASTPRM                         = 00000018             LKBTBL                           00000002 R      02
BSTRING                        = 00000001             LKI$C_LENGTH                   = 00000018
CHECKITEM                        000001F3 R      02   LKI$C_LKBTYPE                  = 00000001
CHECK_SPC                        000002AD R      02   LKI$C_RSBTYPE                  = 00000002
CLU$GL_CLUB                      ********    X   02   LKI$C_WAITING                  = FFFFFFFF
CLUB$L_LOCAL_CSID              = 00000060             LKI$SEARCH_BLOCKEDBY             0000060D RG     02
CSTRING                        = 00000002             LKI$SEARCH_BLOCKING              0000056C RG     02
DYN$C_BUFIO                    = 00000013             LKI$SND_BLKBY                    ********    X   02
EFN                            = 00000004             LKI$SND_BLKING                   ********    X   02
EXE$ALONONPAGED                  ********    X   02   LKI$SND_LOCKS                    ********    X   02
EXE$DEANONPAGED                  ********    X   02   LKI$SND_STDREQ                   ********    X   02
EXE$GETLKI                       00000000 RG     03   LKI$V_SYSNAM                   = 0000001F
EXE$IPID_TO_EPID                 ********    X   02   LKI$_BLOCKEDBY                 = 00000206
EXE$PROBEW                       ********    X   02   LKI$_BLOCKING                  = 00000207
EXE_GETLKI                       00000098 R      02   LKI$_LASTLKB                   = 00000106
GETLKB                           00000493 R      02   LKI$_LASTRSB                   = 00000209
GET_REMLKI                       000001D0 R      02   LKI$_LCKCOUNT                  = 00000205
GRET                             00000177 R      02   LKI$_LCKREFCNT                 = 00000103
IOC$GW_MAXBUF                    ********    X   02   LKI$_LOCKID                    = 00000104
IOSB                           = 00000010             LKI$_LOCKS                     = 00000208
IPL$_ASTDEL                    = 00000002             LKI$_NAMSPACE                  = 00000200
IPL$_SYNCH                     = 00000008             LKI$_PARENT                    = 00000102
ITMLST                         = 0000000C             LKI$_PID                       = 00000100
JIB$L_BYTCNT                   = 00000020             LKI$_REMLKID                   = 00000105
JIB$L_BYTLM                    = 00000024             LKI$_RESNAM                    = 00000201
LCK$CHECK_STALL                  ********    X   02   LKI$_RSBREFCNT                 = 00000202
LCK$COMPAT_TBL                   ********    X   02   LKI$_STATE                     = 00000101
LCK$GL_IDTBL                     ********    X   02   LKI$_SYSTEM                    = 00000204
LCK$GL_MAXID                     ********    X   02   LKI$_VALBLK                    = 00000203
LCK$K_CWMODE                   = 00000002             LKID                           = 00000008
LCK$K_PRMODE                   = 00000003             LKI_ALLOCATE                     000006C9 R      02
LCK$K_PWMODE                   = 00000004             LOCAL_SPACE                    = FFFFFFF8
LIMSG$K_ZERO                   = 00000000             LOCK_INFO                        0000042B R      02
LIMSG$L_LCKCOUNT               = 0000002C             MAXCOUNT                         00000000 R      02
LIMSG$L_RSBREFCNT              = 00000028             MAXSTRUC                       = 00000002
LIMSG$L_STATE                  = 00000024             MAX_LKB_ITEM                   = 00000005
LIMSG$Q_VALBLK                 = 00000030             MAX_RSB_ITEM                   = 00000008
LKB$B_GRMODE                   = 00000035             MOVEIT                           0000025A R      02
LKB$B_RQMODE                   = 00000034             PCB$L_JIB                      = 00000080
LKB$B_STATE                    = 00000036             PCB$L_PID                      = 00000060
LKB$K_CONVERT                  = 00000000             PCB$Q_PRIV                     = 00000084
LKB$K_GRANTED                  = 00000001             PCB$W_ASTCNT                   = 00000038
LKB$K_WAITING                  = FFFFFFFF             PCB$W_GRP                      = 000000BE
LKB$L_CSID                     = 00000058             PR$_IPL                        = 00000012
LKB$L_EPID                     = 00000014             PRV$V_SYSLCK                   = 0000001E
LKB$L_LKID                     = 00000030             PRV$V_WORLD                    = 00000010
LKB$L_PARENT                   = 00000048             PSL$C_EXEC                     = 00000001
LKB$L_PID                      = 0000000C             PSL$C_KERNEL                   = 00000000
```

```
PSL$S_PRVMOD                    = 00000002
PSL$V_PRVMOD                    = 00000016
RESERV                          = 0000001C
RSB$B_RMOD                      = 0000004E
RSB$B_RSNLEN                    = 0000004F
RSB$L_CSID                      = 00000038
RSB$L_CVTQFL                    = 00000018
RSB$L_GRQFL                     = 00000010
RSB$L_WTQFL                     = 00000020
RSB$Q_VALBLK                    = 00000028
RSB$W_GROUP                     = 0000004C
RSB$W_REFCNT                    = 00000040
RSBTBL                            00000026 R      02
SAVED_IPL                       = FFFFFFFC
SCH$CLREF                         ********    X   02
SCH$GL_CURPCB                     ********    X   02
SCH$POSTEF                        ********    X   02
SPC_BLOCKEDBY                     0000035D R      02
SPC_BLOCKING                      00000392 R      02
SPC_LCKCOUNT                      00000332 R      02
SPC_LOCKS                         000003C7 R      02
SPC_NAMSPACE                      00000321 R      02
SPC_PARENT                        000002FD R      02
SPC_PID                           000002D2 R      02
SPC_REMLKID                       00000347 R      02
SPC_STATE                         000002EB R      02
SPC_SYSTEM                        0000030C R      02
SPECIAL                           0000005C R      02
SPECIAL_LEN                     = 0000000A
SS$_ACCVIO                      = 0000000C
SS$_BADPARAM                    = 00000014
SS$_BUFFEROVF                   = 00000601
SS$_EXQUOTA                     = 0000001C
SS$_INSFMEM                     = 00000124
SS$_IVLOCKID                    = 00002124
SS$_IVMODE                      = 00000354
SS$_NOMORELOCK                  = 00000A08
SS$_NORMAL                      = 00000001
SS$_NOSYSLCK                    = 000028F4
SS$_NOWORLD                     = 00002884
SYS$DCLAST                        ********   GX   02
VALUE                           = 00000000
VERIFYLOCKID                      000004E4 R      02


                                                  +----------------+
                                                  ! Psect synopsis !
                                                  +----------------+

PSECT name                        Allocation          PSECT No.  Attributes
----------                        ----------          ---------  ----------
.  ABS  .                         00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
$ABS$                             00000030  (   48.)  01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
WSYSGETLKI                        00000716  ( 1814.)  02 (  2.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
YEXEPAGED                         00000008  (    8.)  03 (  3.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
```

```
                                        +----------------------------+
                                        ! Performance indicators !
                                        +----------------------------+

Phase                        Page faults    CPU Time       Elapsed Time
-----                        -----------    --------       ------------
Initialization                        34    00:00:00.07    00:00:00.38
Command processing                   127    00:00:00.62    00:00:06.40
Pass 1                               488    00:00:19.74    00:00:53.40
Symbol table sort                      0    00:00:02.90    00:00:08.60
Pass 2                               289    00:00:05.01    00:00:11.37
Symbol table output                   19    00:00:00.16    00:00:00.37
Psect synopsis output                  2    00:00:00.03    00:00:00.03
Cross-reference output                 0    00:00:00.00    00:00:00.00
Assembler run totals                 961    00:00:28.53    00:01:20.55

The working set limit was 2100 pages.
113868 bytes (223 pages) of virtual memory were used to buffer the intermediate code.
There were 100 pages of symbol table space allocated to hold 1809 non-local and 102 local symbols.
1661 source lines were read in Pass 1, producing 22 object records in Pass 2.
39 pages of virtual memory were used to define 38 macros.

                                        +----------------------------+
                                        ! Macro library statistics !
                                        +----------------------------+

Macro library name                               Macros defined
------------------                               --------------
_$255$DUA28:[SHRLIB]CLUSTER.MLB;1                            1
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                              18
_$255$DUA28:[SYSLIB]STARLET.MLB;2                           13
TOTALS (all libraries)                                      32

1957 GETS were required to define 32 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:SYSGETLKI/OBJ=OBJ$:SYSGETLKI MSRC$:SYSGETLKI/UPDATE=(ENH$:SYSGETLKI)+EXECML$/LIB+SHRLIB$:CLUSTER/LIB
```

0385 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY


SYSGETSYI
LIS

SYSGETJPI
LIS

SYSIMGFIX
LIS

SYSGETTIM
LIS

SYSGETLKI
LIS

SYSGETMSG
LIS

SYSIMGACT
LIS